HP reference configuration for Scalable Warehouse Solution for Oracle: HP DL980 G7 and P2000 G3 MSA

Technical white paper

Table of contents

Executive summary 2

Introduction 2
- Important points and caveats 3

Solution criteria 3
- Recommended configurations 5

System/Environment setup 6
- Storage configuration details 6
- Server configuration details 12
- Linux configuration details 17
- Oracle Database configuration details 18

Measured performance 20
- Results of Parallel Query Execution testing 20

Best practices when deploying 26
- Linux operating system 26
- DL980 server best practices 26
- Oracle Database best practices 26
- Storage best practices 27

Bill of materials 28

Implementing a proof-of-concept 30

Appendix 1: BIOS and OS settings 31

Appendix 2: Oracle parameter settings 33

Appendix 3: Example multipath.conf for P2000 array 35

Appendix 4: Oracle Advanced Compression 39

For more information 40

Executive summary

The HP Scalable Warehouse Solution for Oracle on Hewlett-Packard (HP) servers, storage and networking products provides a prescriptive approach for balancing server, storage, network and software configurations for architecting Oracle data warehouse solutions. The reference architectures provide server and storage guidance for various data warehouse workloads – giving you the most efficient configuration for your solution, saving you time and cost in choosing the right technology, and giving you peace of mind that the right platform and architecture is in place.

Target audience: The target audience for this document consists of IT planners, architects, system administrators, DBAs, CIOs, CTOs, and business intelligence (BI) users with an interest in options for their BI applications and in the factors that affect those options.

This white paper describes testing performed in May-September 2011.

If you need help with a specific Oracle solution or prefer a solution design or sizing based on your requirements please contact your local HP reseller, HP sales representative, or the HP Oracle Solution Center in your region.

Introduction

This document is a reference configuration for the HP Scalable Warehouse Solution for Oracle, which describes a repeatable architectural approach for implementing a scalable model for a symmetric multiprocessor (SMP)-based Oracle data warehouse. The end result of the process described in this guide represents a recommended minimal Oracle database configuration, inclusive of all the software and hardware, required to achieve and maintain a baseline level of "out of box" scalable performance when deploying Oracle data warehousing sequential data access workload scenarios versus traditional OLTP random I/O methods.

This document provides specific details about the testing, configuration and bill of materials for such reference architecture, based on the HP ProLiant DL980 G7 and the HP P2000 G3 Modular Smart Array (P2000 G3 MSA). This configuration is targeted at a data warehouse / data mart environment with scan rate requirements of 18GB/sec. It is optimized at 19TB of usable user data (RAW) capacity.

Some attributes of this solution includes:

- Single scale up or multiple scale out server database applications
- High sustained sequential read I/O throughput (high performance)
- High Availability storage redundancy
- Server redundancy (through Oracle Data Guard or Real Application Cluster)
- Large capacities of usable storage
- Performance scalable solution

HP Scalable Warehouse Solution for Oracle:

- Delivering leading headroom and expandability for x86 virtualization and enterprise applications
- Efficiently use compute resources with HP's market leading 8 socket architecture
  - Smarter CPU caching strategy, improves CPU utilization and performance
  - Flexible expansion of CPU, memory and I/O capacity to grow with your needs
  - Single 8 processor server to multiple server growth path to support large database environments
- Faster design to implementation, with HP and key industry partner solution stacks extended to address scale-up x86 environments
  - Collaborative partnerships: Microsoft®, Red Hat, SUSE, Oracle and VMware
  - Operating system and virtualization software enhanced to support large scale-up x86 environments
  - HP Management software to help reduce and manage complexity
  - HP consulting services for implementing BI strategies
- Speed your way to better business outcomes with
  - HP's in-depth Solution and Services expertise
  - HP's broad portfolio of complementary products

Important points and caveats

- The configuration described here is exclusively designed for, and is applicable to, sequential data workloads or limited mixed workloads. Use of this approach on other workload types would need to be tested and may yield configurations that are effective for other solutions.
- **ALL** recommendations and best practices defined in the reference architecture guide must be implemented in their entirety in order to preserve and maintain the sequential order of the data and sequential I/O against the data.

A complete Oracle database DBMS (Database Management System) configuration is a collection of all the components that are configured to work together to support the database application. This includes the physical server hardware (with its BIOS settings and appropriate firmware releases), memory, CPU (number, type, clock and bus speed, cache and core count), operating system settings, the storage arrays and interconnects, disk (capacity, form factor and spindle speeds), database, DBMS settings and configuration, and even table types, indexing strategy, and physical data layout.

The primary goal of the HP Scalable Warehouse Solution for Oracle, which is also a common goal when designing most data center infrastructures, is a balanced configuration where all components can be utilized to their maximum capability. Architecting and maintaining a balance prevents over subscribing certain components within the stack to a point where the expected performance is not realized; understanding the performance limits of your configuration can help prevent wasted cost for components that will never realize their potential due to other constraints within the stack.

Solution criteria

The HP Scalable Warehouse Solution for Oracle reference configurations are built on the DL980 G7 highly scalable HP ProLiant server platform, each targeting a different tier of an Oracle Data Warehousing solution. The HP Scalable Warehouse Solution for Oracle architectures gain greater throughput and scale by using the following approach:

- Targeting query workloads patterned for large sequential data sets rather than small random read/write data transactions
- Optimizing rapid data reads and query aggregations

This configuration leverages the HP P2000 G3 MSA FC Dual Controller SFF array, which allows for dual reads when drives are mirrored. For sequential data reads from data warehouse queries, this capability enables tremendous throughput for the storage system – up to 18,000 MB/sec. The HP Scalable Warehouse Solution for Oracle approach, and supporting storage array architecture, is optimized for sequential reads. To support a non-optimized, random I/O data warehousing workload, up to 2 to 3 times the number of drives would be required to achieve the same throughput. Another approach for random I/O data warehousing is to look at HP High Performance Database Solution for Oracle or HP Data Accelerator Solution for Oracle.

Tables 1 through 4 below list the supported Intel® Xeon® processors, memory DIMMs, and PCI expansion slots for the ProLiant DL980 G7 server. This information is included to help determine how the recommended configurations can be modified to support different workload sizes or user combinations.

Table 1. Supported E7 Family Processors

Processor Type Intel Xeon	Cores per Processor	Max Cores in a DL980 G7
E7-4870 (30MB Cache, 2.4GHz, 130W, 6.4 GT/s QPI)	10	80
E7-2860 (24MB Cache, 2.26GHz, 130W, 6.4 GT/s QPI)	10	80
E7-2850 (24MB Cache, 2.00GHz, 130W, 6.4 GT/s QPI)	10	80
E7-2830 (24M Cache, 2.13 GHz, 105W, 6.4 GT/s QPI)	8	64
E7-4807 (18M Cache, 1.86GHz, 95W, 4.8 GT/s QPI)	6	48

Note
The Intel Xeon processor E7 series supports Hyper-Threading (HT). HT is recommended and was tested in our configuration. However it is good practice to test HT with your particular application.

Table 2. Supported Memory DIMMs

Memory Kits	Rank
HP 4GB 1Rx4 PC3-10600R-9	Single
HP 4GB PC3L-10600R-9 512Mx	Single
HP 8GB 2Rx4 PC3-10600R-9	Dual
HP 8GB 2Rx4 PC3L-10600R-9, 512Mx RoHS	Dual
HP 16GB 2Rx4 PC3L-10600R-9	Dual
HP 16GB 4Rx4 PC3-8500R-7	Quad

PC3L = low voltage memory

Table 3 represents the minimum, middle, and maximum memory combinations possible for the 4, 8, and 16 GB memory kits available for the DL980 G7 servers. Memory is installed into cartridges; each cartridge supports a minimum of 2 DIMMS and a maximum of 8 DIMMS. For best performance, HP recommends the use of dual or quad rank memory DIMMs.

Table 3. Minimum, middle, and maximum memory for 4 and 8 processor configurations

Number of CPUs	Memory Density (GB)	Total Memory Cartridges	Min Memory (GB)	Mid Memory (GB)	Max Memory (GB)
4	4	8	64	128	256
4	8	8	128	256	512
4	16	8	256	512	1024
8	4	16	128	256	512
8	8	16	256	512	1024
8	16	16	512	1024	2048

Note
Max memory depends on the number of processors configured. Four/eight processor configurations support up to 1TB/2TB of memory, respectively. However Red Hat Enterprise Linux 5.x only supports up to 1TB of memory. Red Hat Enterprise Linux 6.x supports the maximum memory configuration of the DL980 (though official Oracle support for that platform was not announced as of the initial publication of this document).

Table 4. DL980/P2000 supported Expansion Slot Configurations

Expansion Slots
• Standard Main I/O with 5 Gen2 slots: (3) x4 PCI-Express; (2) x8 PCI-Express • PCIe Option with 6 slots: (1) x4 Gen1 PCI-Express; (1) x4 Gen2 PCI-Express (4) x8 Gen2 PCI-Express • Low Profile Expansion Option with 5 Gen2 slots; (1) x4 PCI-Express; (4) x8 PCI-Express

Recommended configurations

The detailed information for these recommended configurations includes the server, number and type of processors, memory, internal and external storage. The configurations were evaluated for a given workload: concurrent users, I/O throughput, and database size. The configurations are based on testing done in HP's Oracle integration lab using HP DL980 G7 ProLiant servers running Red Hat Enterprise Linux (RHEL) and Oracle 11gR2 Enterprise Edition.

The configurations were determined based on the following utilization criteria:

- CPU utilization up to 90% at the target workload
- Process Global Area (PGA) cache hit ratio 99% or higher with very low counts of 1 or multipass PGA executions
- Disk I/O activity reflects a read/write ratio of approximately 95/5

The focus of this reference configuration document is the data mart or data warehouse type configuration, based on the ProLiant DL980 G7.

Table 5. Recommended Configuration metrics and storage type options

Number	HP ProLiant Server Model	SAN	Storage Drive Count	Storage Type	Database Size TB	I/O throughput MB/sec
1	DL980 G7 (8p/80c)	(12) HP P2000 G3 MSA	(288) 146GB 6G 15K SAS SFF	HDD	19	18,000

Table 6 outlines the server configuration details for this platform.

Table 6. ProLiant DL980 G7 configuration details

Model	ProLiant DL980 G7
CPU	(8) Ten-Core Intel Xeon Processors Model E7-4870 (30MB Cache, 2.4GHz, 130W, 6.4 GT/s QPI)
Number Cores	80
PCI-E Slots	(10) x8, (2) x4, all FL/FH
Drives	(4) HP 146GB 6G SAS 15K rpm SFF (2.5-inch) DP HDD
Storage Controller	HP Smart Array P410i/1GB FBWC
Host Bus Adapters	(12) HP 82Q PCI-e 8Gb/s FC Dual Port HBA (QLogic)
Network Adapters	One HP NC375i Quad port Gigabit Server Adapters (four ports total)
RAM	1024GB PC3-10600R-9 expandable to 2TB

Note that the recommended 1024GB is the minimum RAM for this reference configuration. Generally, for Oracle Data Warehousing environments, as workload demands grow, increasing RAM provides performance benefits (requires RHEL 6.x). If the workload consists of a large number of small-to-medium sized queries hitting the same data (for example, last week's sales query), performance and throughput can be increased by caching results in memory.

Additional network capacity can be added to support larger numbers of client connections however most DW workloads have very few client connections.

System/Environment setup

Storage configuration details

Internal storage

The DL980 G7 recommended configurations use four internal drives configured with RAID1 for the OS and 11gR2 software. The server supports up to eight internal drives, so additional drives can be added for staging data, logs, or other requirements.

Table 7. Internal storage controllers

Database Server	Internal Storage Controller
DL980 G7	• HP Smart Array P410i/Zero Memory Controller (RAID 1) • HP 146GB 6G SAS 15K rpm SFF Hard Drives • Available upgrades: Battery kit upgrade, 512MB Flash Backed Write Cache (FBWC), 1G Flash Backed Write Cache, and Smart Array Advanced Pack (SAAP)

External storage

The HP Scalable Warehouse Solution for Oracle reference configuration is based on a model of a core-balanced architecture which allows for a configuration that is initially balanced and provides predictable performance. This core balanced architecture is sized for an optimum number of storage components to drive a certain number of CPU cores. Each CPU core has a maximum throughput of information that can be processed. If we take the per core rate and multiply times the number of CPU cores, we then have the processing rate for the entire system. To ensure balanced performance we must ensure that there are minimal bottlenecks in the system that would prevent the CPUs from reaching their maximum throughput.

It is also possible to increase storage capacity without adding more CPU cores by using larger drives; however increasing the capacity beyond the optimized configuration will not increase performance. Important to note is that adding more data but still querying the same sized ranges per query will not decrease performance. Table 8 summarizes the configuration specifics and capabilities of the fibre channel version of the P2000, the P2000 G3 MSA FC.

Table 8. P2000 G3 MSA FC storage configuration details

Model	(12) P2000 G3 MSA FC SSF Array with dual read controllers
Array Configuration	Cache Optimization – Standard Read Ahead Size – 2MB Cache Write Policy – write back
Drives	(288) 146GB 6G 15K SFF SAS disks
Drive layout	(288) drives all for user data (24 per enclosure), configured as (48) 6-disk RAID10 vdisks presented as host LUNs 64KB Chunk size used (4) 400GB volumes per P2000 array
User Database Space	19 TB
DB Temp Space	500GB
Redo Log Capacity	20GB
Secondary Staging Space (10%)	1.9 TB
Measured I/O Rate	18GB/Sec

Note
To locate and download the latest software and firmware update for your P2000, go to http://www.hp.com/go/p2000. Select **Models** – select your product **P2000 G3 MSA Fibre Channel Dual Controller SFF Array System** – select **HP Support & Drivers**, then select **Download drivers and software**. Also you can get support information for any HP products by going to http://www.hp.com/go/support.

The P2000 Storage Management Utility (SMU) is a web-based application for configuring, monitoring, and managing the storage system. Within the SMU the Provisioning Wizard will help you create the vdisk layout with volumes and will map the volumes to the DL980 server. On the server itself, Oracle Automatic Storage Management (ASM) will allocate the LUNs to a diskgroup. Before using this wizard, read the documentation and SMU reference guidelines to learn about vdisks, volumes, and LUN mapping. A command line interface is also available and can be used for scripting and bulk management.

A vdisk is a "virtual" disk that is composed of one or more physical hard drives, and has the combined capacity of those disks. The number of disks that a vdisk can contain is determined by its RAID level. In a dual-controller P2000 system, when a vdisk is created the system automatically assigns the owner to balance the number of vdisks each controller owns. Typically it does not matter which controller owns a vdisk. In a dual-controller system, when a controller fails, the partner controller assumes temporary ownership of the failed controller's vdisks and resources. When a fault-tolerant cabling configuration is used to connect the controllers to FC SAN switches and hosts, both controllers' LUNs are accessible through the partner.

When you create the vdisks select the 64KB chunk size. The chunk size is the amount of contiguous data that is written to a disk before moving to the next disk. The 64KB chunk size provided the best overall performance in our reference configuration testing. That means the requests would be spread evenly over all of the disks, which is good for performance.

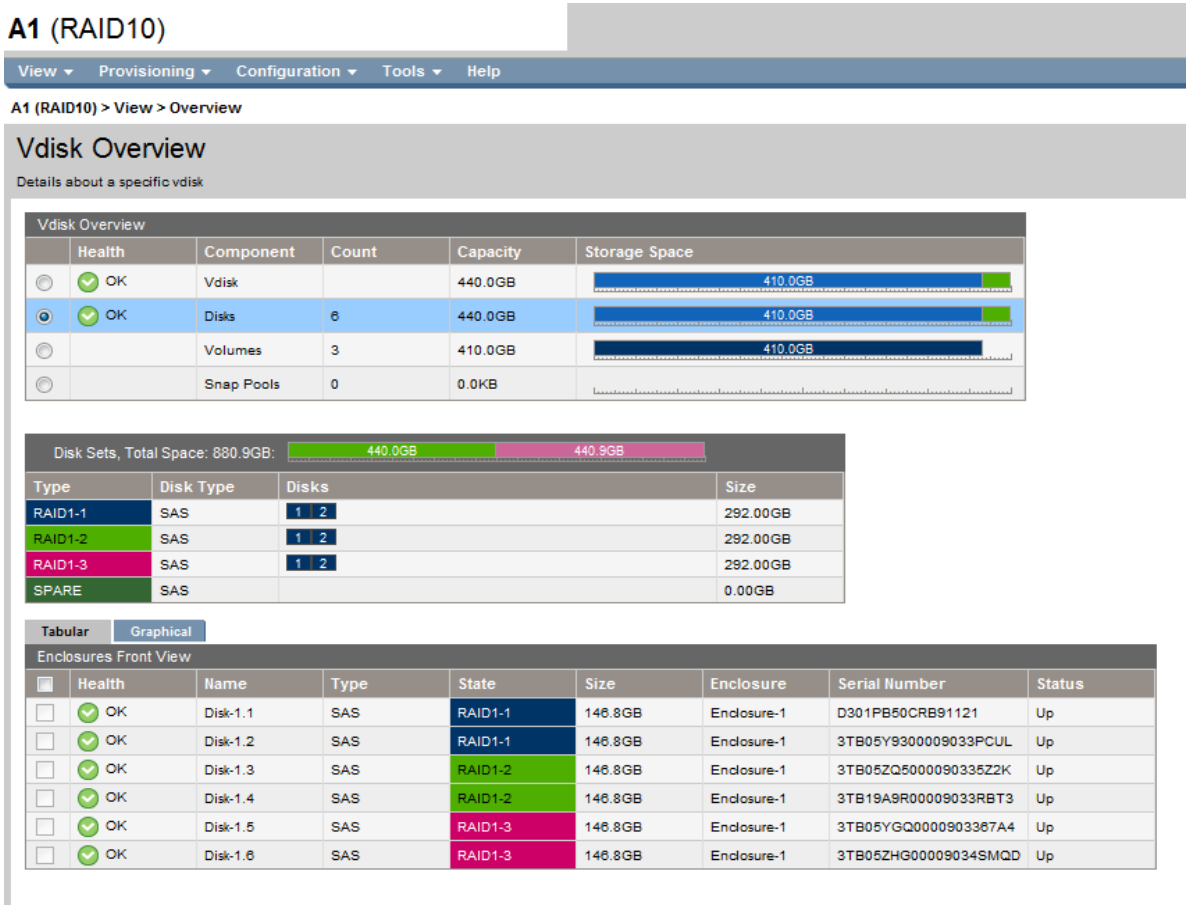
When you create a vdisk you also create volumes within it. A volume is a logical unit number (LUN) of a vdisk, and can be mapped to controller host ports for access by hosts. A LUN identifies a mapped volume to the DL980. The storage system presents only volumes, not vdisks, to hosts.

Some best practices to keep in mind for creating vdisks include:

- To maximize capacity, use disks of similar size.
- For greatest reliability, use disks of the same size and rotational speed.
- The optimal configuration for the tested BI workload was to create 4 vdisks of 6 physical drives each for every 24-disk P2000 array.
- For maximum use of a dual-controller system's resources, the vdisks for each array should be evenly divided between the controllers.
- For a fault-tolerant configuration, configure the vdisks with write-back caching.

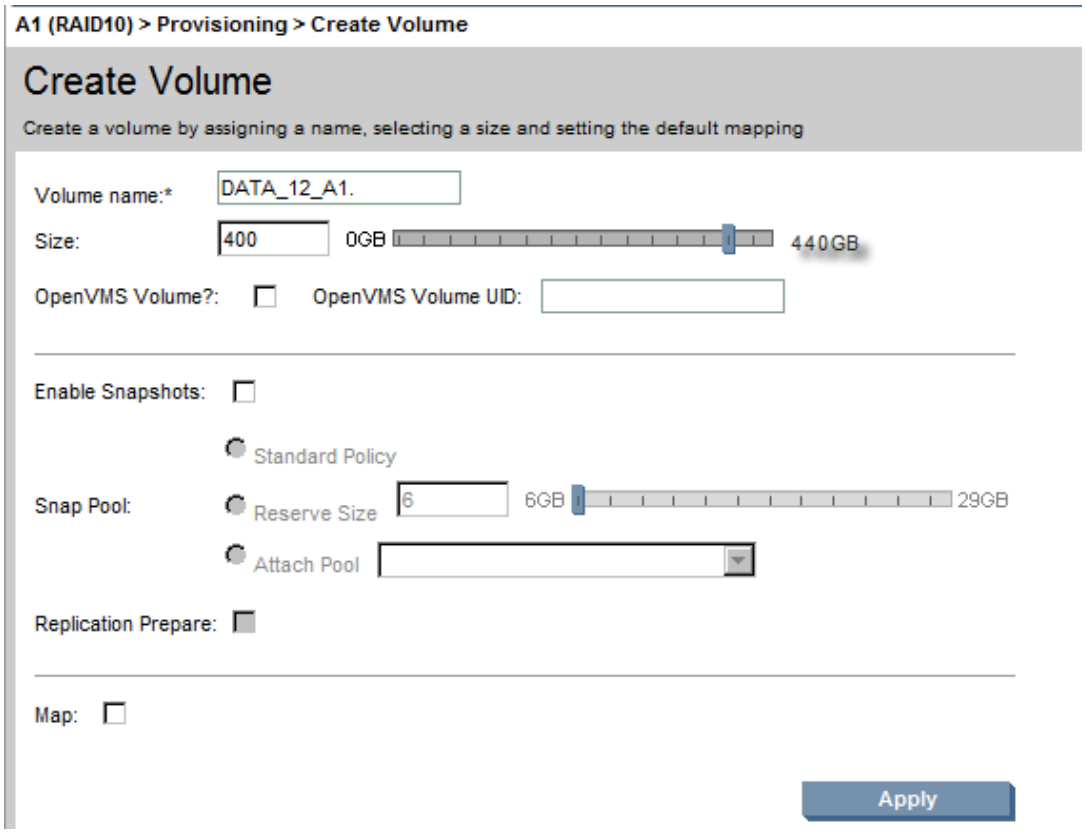
For our reference configuration, each P2000 array was divided into four vdisks with six 146GB drives each. The vdisks were configured for RAID10 and a 64KB chuck size. The vdisks were named A1, A2, B1, B2 so as to simplify the mapping of the vdisks to the fibre channel ports of the P2000 controllers. A single 400GB volume was created from each vdisk to be used for the Oracle database (see example in figure 2). This provided a total of 48 times 400GB volumes (19TB database storage) for presentation to the DL980 where Oracle ASM is used to layout the filesystem so we can then create the database. The remaining 1.9TB of storage space (40 GB unallocated from each vdisk) was reserved for flat files, staging or other application requirements. Figure 1 shows an example of one of the four vdisk configurations with the name A1 using 6 disks striped with mirroring within a single P2000 array. Another way of creating the storage layout is by using command line and scripts.

Figure 1. P2000 Provisioning Wizard with details of one vdisk within a single P2000



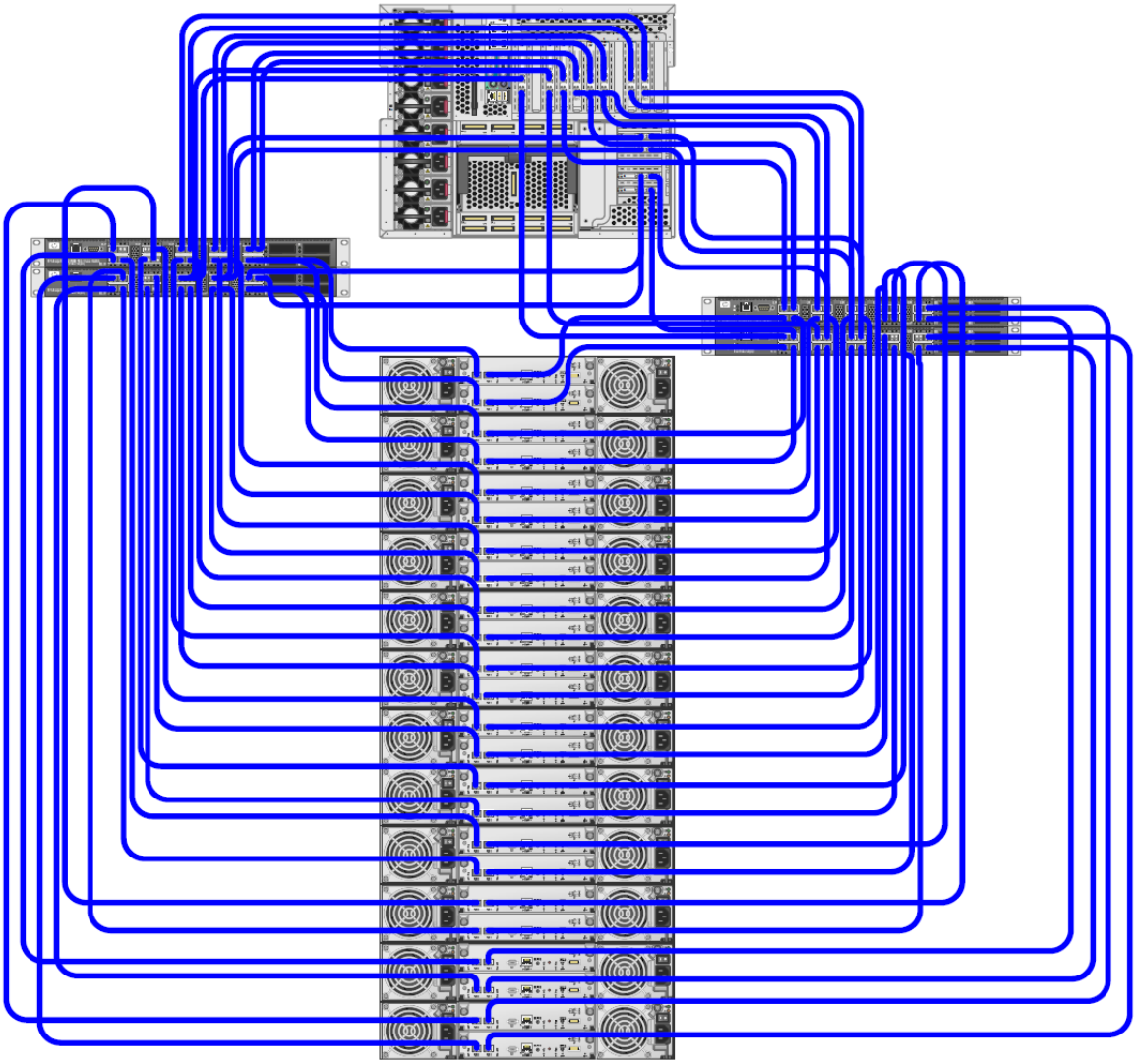
Properties for A1	
Property	**Value**
Health	OK
Health Reason	Virtual-disk is fault tolerant.
Name	A1
Size	439.9GB
Free	29.9GB
Current Owner	A
Preferred Owner	A
Serial Number	00c0ffdadaf700007d04f94c00000000
RAID	RAID10
Disks	6
Spares	0
Chunk Size	64k
Created	2010-12-03 14:53:49
Minimum Disk Size	146.6GB
Status	FTOL
Current Job	
Drive Spin Down Vdisk Enable	Disabled

Figure 2. Creating a single 400GB data volume from vdisk A1. Explicit mapping done later



A P2000 enclosure has two controllers each with two FC ports. These ports are labeled A1 and A2 for controller A and B1 and B2 for controller B. Fibre Channel connectivity was accomplished with the two FC ports on each controller of the P2000 going to a separate FC SAN switch and then each dual port FC card on the DL980 is connecting to the two separate SAN switches. This FC port mapping of each P2000 controller to separate FC HBAs and HP SN6000 Stackable 8 Gb 24-port Dual Power Fibre Channel switches provides for failover redundancy. See Figure 3 for connectivity diagram.

Figure 3. DL980, P2000 and SN6000 FC Switch connectivity diagram.



Once the vdisks and volumes are created for each P2000 the next step is to explicitly map the volume to the controller host ports for access by the host. Both controllers share one set of LUNs so if a controller fails the other controller can access all LUNs on the array. Each 400GB volume was assigned as an explicit type LUN to one of the DL980 FC HBA Host IDs as read-write for each of the two controllers in the P2000 enclosure; so in our example using the storage management GUI we select our volume DATA_12_A1 (data volume for array 12 on Controller A1 port) and then select Explicit Mappings to bring up the screen shown in figure 4, where controller port A1 and LUN 1 are explicitly assigned to a FC HBA ID and controller B1 and LUN 2 are assigned to a second FC HBA ID as a secondary path. This allows us to distribute the volumes evenly over the 24 FC HBAs and setup OS multipathing for primary and failover paths for redundancy.

We would in turn do this for the remaining 47 vdisk volumes. The 24 FC HBA ports on the DL980 are identifiable by "50014" as the beginning of the Host ID (WWPN) in our list below.

Figure 4. Specifying the explicit mapping to DL980 FC HBA WWPN IDs for access to the volume DATA_12_A1

Maps for DATA_12_A1

Type	Host ID	Name	Ports	LUN	Access
Explicit	50014380062ca7f6		A1	1	read-write
Default	50014380062c9fb2				not-mapped
Default	50014380062ca6a0				not-mapped
Default	50014380062ca6a2				not-mapped
Default	50014380062ca87c				not-mapped
Default	50014380062ca87e				not-mapped
Default	50014380062c9fb0				not-mapped
Default	500143800566585c				not-mapped
Default	500143800566585e				not-mapped
Default	50014380062ca878				not-mapped
Default	50014380062ca87a				not-mapped
Default	50014380062ca7f4				not-mapped
Default	21fd00051ef34553				not-mapped
Default	21fd00051e7357de				not-mapped
Default	50014380062c9d52				not-mapped
Explicit	5001438005665a84		B1	2	read-write
Default	50014380062ca880				not-mapped
Default	50014380062ca882				not-mapped
Default	5001438005665792				not-mapped
Default	50014380062ca2e6				not-mapped
Default	50014380062c9d50				not-mapped
Default	5001438005665a86				not-mapped
Default	50014380056656d4				not-mapped
Default	50014380056656d6				not-mapped
Default	5001438005665790				not-mapped
Default	50014380062ca2e4				not-mapped
Default	21fd00051e735e9b				not-mapped
Default	21fd00051eed6148				not-mapped

In table 9 is a summary of what the volume mapping looked like for a single P2000 array. Since we have a total of 48 volumes and 24 FC HBA ports we mapped two volumes to a primary and secondary FC HBA port spread across two different controller ports and LUN numbers.

Table 9. Explicit mapping of the four 400GB volumes in P2000 enclosure 12 mapped to DL980 FC HBAs, Controller ports and LUN numbers.

Volume Name	HOST ID (FC HBA)	Controller Port	LUN #
DATA_12_A1	50014380062ca7f6 5001438005665a84	A1 B1	1 2
DATA_12_A2	50014380062c9fb2 50014380062c9d52	A2 B2	1 2
DATA_12_B1	50014380062ca7f6 5001438005665a84	A1 B1	2 1
DATA_12_B2	50014380062c9fb2 50014380062c9d52	A2 B2	2 1

This configuration may appear complex, but the model is really simple, two dedicated FC paths for each vdisk to the DL980 so as to optimize the bandwidth between the Linux operating system and the spinning hard drives. With 48 vdisks, effectively each of the 24 FC ports on the DL980 was dedicated to two vdisks (12 disk spindles). This allowed us to maximize the I/O rates adequately for this solution.

Server configuration details

Reducing the cost of BI implementations, while improving the efficiency, are current priorities for many data center managers. Many enterprises are achieving these goals by migrating their core BI data warehouse (DW), data marts, operational data stores (ODS), and BI applications off of expensive and proprietary Oracle-Sun SPARC and IBM POWER platforms and onto standards-based Intel servers. Also occurring in many enterprises is the reduction or elimination of these costly BI silos by adopting a common integrated infrastructure. Reducing time to implementation of a BI solution is a critical success factor and a common, integrated infrastructure improves the time to solution. The HP ProLiant DL980 G7 server with Intel Xeon processor E7 series has been designed with the enterprise-wide BI workload in mind.

The DL980 is well situated to solve the performance and fault-tolerant characteristics inherently desired in a BI workload infrastructure. Supporting a performance optimized BI solution, the DL980:

- Leverages enterprise functionality from HP Integrity enterprise servers for both performance and reliability
- Contains up to 80 high-power Intel Xeon E7 processor cores
- Provides up to 160 logical CPUs with Intel Hyper-Threading Technology to facilitate query parallelism which can help in BI type workloads
- Provides enterprise-level parallel access to storage with a measured 25GB/sec. sustained I/O throughput rate (as measured in the 3TB TPC-H test) from up to 16 PCIe expansion slots
- Provides up to 2TB of memory to facilitate large in-memory processing

To extend processing capabilities even further, it is recommended based on our lab testing to enable Intel Hyper-Threading and each core will appear as two logical processors to the OS. Likewise, a single physical processor with 10 cores appears as 20 functional processors to both the OS and the Oracle database.

HP ProLiant DL980 G7 servers ship with Hyper-Threading enabled by default. Depending on the workload, Hyper-Threading can increase system performance by up to 40% (20% is typical). This increase is seen most often in highly parallelized workloads. Serial workloads (where progress in some application workstreams is heavily dependent on a few serial tasks or there is some explicit contention on system resources) may experience a decrease in performance when Hyper-Threading is enabled. Customers should always test their particular workload with and without Hyper-Threading before committing to its use.

In addition, the DL980 contains HP's PREMA Architecture, which brings many performance and scalability benefits for processing the BI workload. Large numbers of processors in x86 servers typically create an inter-processor communication overhead. To solve this issue for x86 servers, HP looked to the design of our higher-end ccNUMA

scale-up servers. At the core of the HP PREMA Architecture is a node controller ASIC, derived from Intel technology powering the HP Integrity Superdome 2.

The HP PREMA Architecture provides these particular benefits for processing and data-intensive BI workloads:

- Smart CPU caching: Achieves up to 20% better processor scaling than competitive 8 socket systems by reducing processor overhead *
- Resilient system fabric: Provides 50% more interconnect capacity and dynamic traffic routing capability for improved performance in highly concurrent, highly parallel BI workload environments*

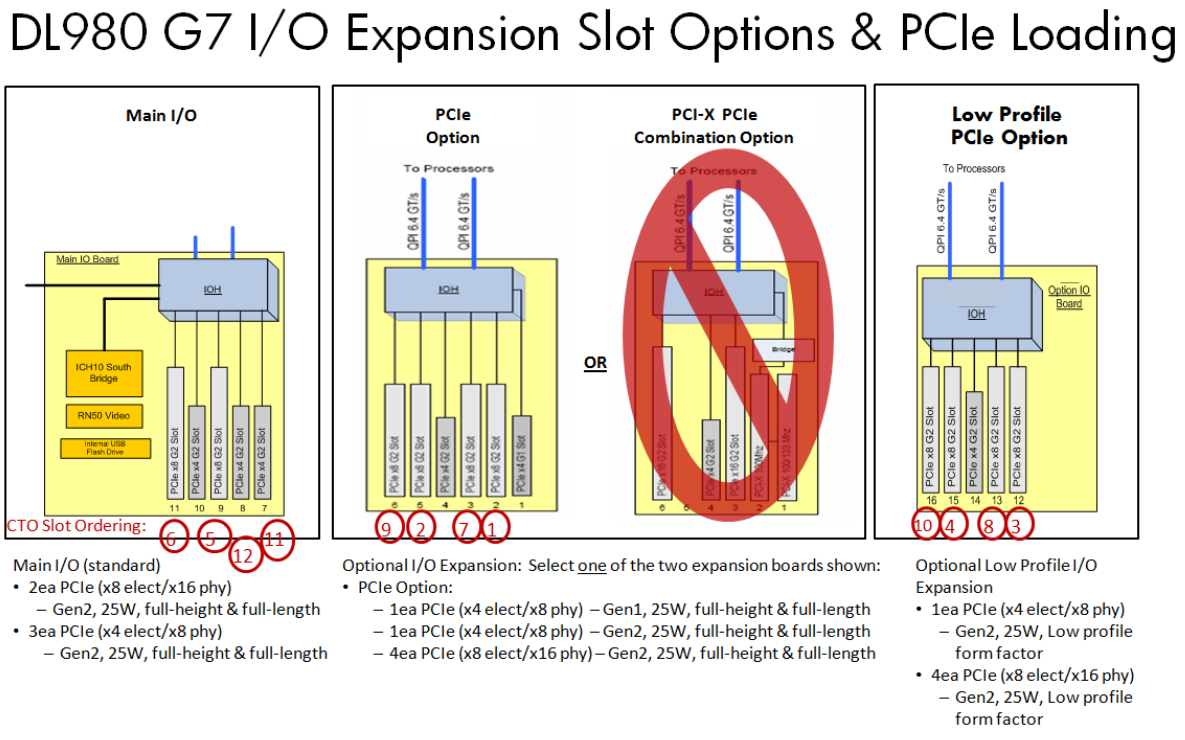
The DL980 provides an excellent platform for scaling data warehouse or data mart solutions. The following information provides detailed setup recommendations used in our reference configuration.

When setting up the DL980 please make sure to review the BIOS settings listed in Appendix 1.

The recommend layout of the fibre channel cards in the DL980 is shown in figure 5. There were (10) PCIe x8 slots and (2) PCIe x4 slots used. For an 8Gb DP FC HBA a PCIe Gen2 x4 slot will provide plenty of bandwidth and will not be a performance bottleneck.

* Based on HP internal engineering benchmarks.

Figure 5. HP Scalable Warehouse Solution for Oracle – DL980 recommended 8Gb FC HBA Slot Loading


DL980 G7 I/O Expansion Slot Options & PCIe Loading
Main I/O
PCIe Option
PCI-X PCIe Combination Option
Low Profile PCIe Option
To Processors
QPI 6.4 GT/s
IOH
Main IO Board
ICH10 South Bridge
RN50 Video
Internal USB Flash Drive
PCIe x8 G2 Slot
PCIe x4 G2 Slot
OR
Option IO Board
CTO Slot Ordering:
6 5 12 11
9 2 7 1
10 4 8 3
Main I/O (standard)
• 2ea PCIe (x8 elect/x16 phy)
– Gen2, 25W, full-height & full-length
• 3ea PCIe (x4 elect/x8 phy)
– Gen2, 25W, full-height & full-length
Optional I/O Expansion: Select one of the two expansion boards shown:
• PCIe Option:
– 1ea PCIe (x4 elect/x8 phy) – Gen1, 25W, full-height & full-length
– 1ea PCIe (x4 elect/x8 phy) – Gen2, 25W, full-height & full-length
– 4ea PCIe (x8 elect/x16 phy) – Gen2, 25W, full-height & full-length
Optional Low Profile I/O Expansion
• 1ea PCIe (x4 elect/x8 phy)
– Gen2, 25W, Low profile form factor
• 4ea PCIe (x8 elect/x16 phy)
– Gen2, 25W, Low profile form factor

Note

Even though internal testing was done using a DL980 with 64 cores and 12 FC HBA cards, from an HP support requirement there is currently a support limit of 11 PCIe FC HBA cards using either QLogic or Emulex FC 8Gb/s HBA cards. With the increase in processor cores supported on the DL980, more FC IO cards may improve the overall throughput of the server. HP is working on enabling more than 11 FC cards on the DL980 that will be available by the end of 2012. QLogic and Emulex vendors are working on a BIOS upgrade to increase the PCIe FC card limit beyond 11.

However there is a workaround for QLogic DP FC cards as long as you are not booting from SAN. Ensure DL980 BIOS P66 06/24/2011 or greater is installed.

Please check SPOCK for availability details.

See http://www.hp.com/storage/spock (requires an HP Passport account).

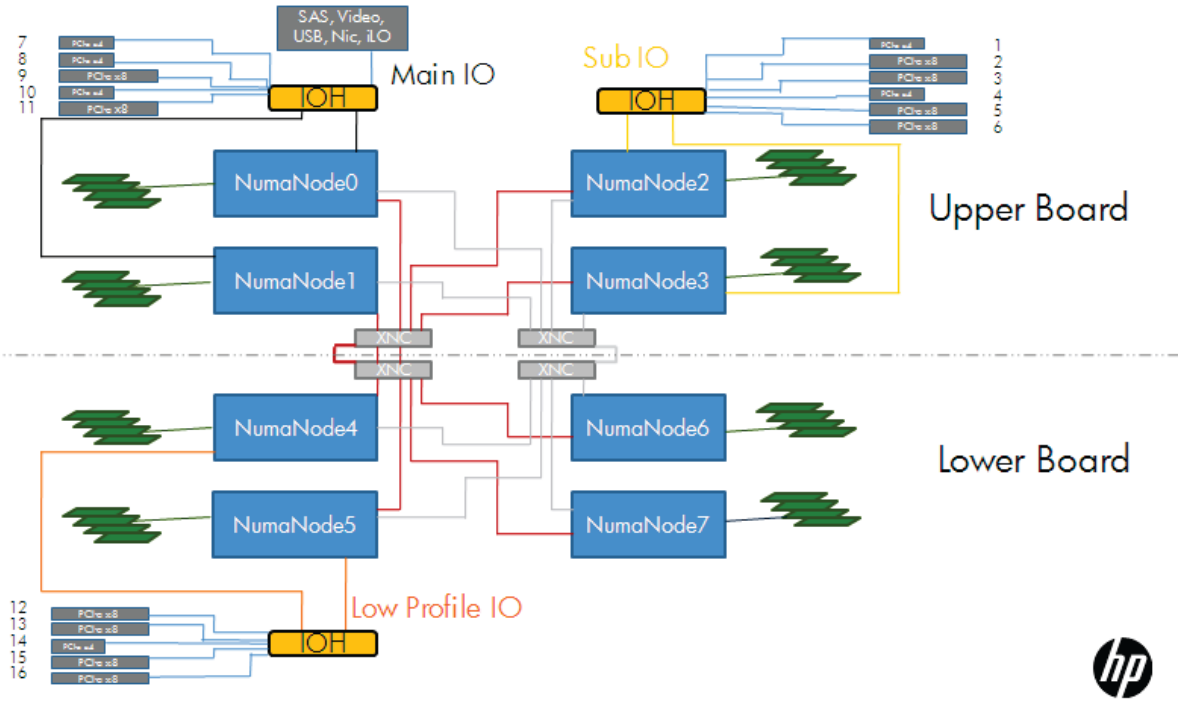
It is best practices to spread the workload evenly over all I/O subsystems. The DL980 has 3 I/O hub (IOH) subsystems, and with the PCIe I/O trays used, slots 7-11 are on the Main IOH, slots 1-6 on the first optional IOH and slots 12-16 on the 2nd optional IOH (low profile).

The best I/O layout specified was:

- IOH 1 – slots 7, 8, 9, 11 (there are only two x8 slots on this subsystem as PCI lanes are used for onboard I/O such as network card NC375i, video, etc.)
- IOH 2 – slots 2, 3, 5, 6
- IOH 3 – slots 12, 13, 15, 16

The workload distribution over the IOHs is more important than the slots, as the IOH goes directly to a pair of CPU sockets to handle the IRQs, etc. The IOH 1 and 2 are connected to the upper CPU board while IOH 3 is connected to the lower CPU board. The following diagram illustrates the physical layout of I/O slot locations and other components, relative to the DL980 processor boards.

Figure 6. I/O slots vs. processor boards connectivity



Note
Do not use slot 1 for any HP FC PCIe HBA Cards due to low I/O performance (PCI-x 100 MHz).

Make sure the FC cards have up-to-date firmware installed. Out of date firmware on Fibre Channel cards may cause performance problems under the extremely high I/O rates possible with this configuration.

Oracle Automatic Storage Management (ASM) I/O configuration and disk layout

Oracle 11gR2 ASM was used for the database storage. Oracle ASM is a volume manager for Oracle database files that supports single-instance Oracle Database and Oracle Real Application Clusters (Oracle RAC) configurations. Oracle ASM uses disk groups to store data files; an Oracle ASM disk group is a collection of disks/LUNs that Oracle ASM manages as a unit. Within a disk group, Oracle ASM exposes a file system interface for the Oracle database files to access. The content of files that are stored in a disk group is evenly distributed to eliminate hot spots and to provide uniform performance across the disks. The ASM I/O performance can approach the performance of raw devices.

The Oracle ASM volume manager functionality can provide server-based mirroring options. However we used the external redundancy setting since the P2000 storage subsystem was configured to mirror the storage volumes. All 48 host LUNs from the P2000 were presented to ASM using external redundancy.

Every Oracle ASM disk is divided into smaller increments called allocation units (AU). An allocation unit is the fundamental unit of allocation within a disk group. A file extent consists of one or more allocation units. An Oracle ASM file (control files, data, index, redo log, etc.) consists of one or more file extents. When the disk group is created, you can set the Oracle ASM allocation unit size with the AU_SIZE attribute. In 11gR2 the Oracle ASM allocation unit size is no longer a hidden parameter and can be set to the following sizes 1, 2, 4, 8, 16, 32, or 64

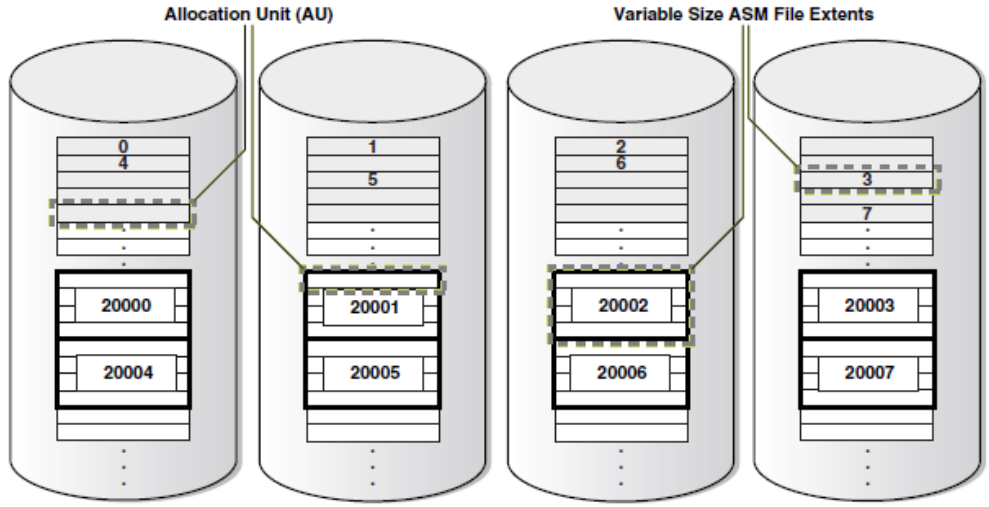
MB, depending on the specific disk group compatibility level. Larger AU sizes typically provide performance advantages for data warehouse applications that use large sequential reads.

Oracle ASM 11gR2 introduced the concept of variable extents, which adds an extra layer of complexity when determining the optimal AU size. The extent size of a file varies as follows:

- Variable ASM extents used for Allocation Units
- First 20,000 (0-19,999) AUs the extents size = 1AU
- Next 20,000 (20,000 – 39,999) AUs the extent size = 8*1AUs
- Then 40,000+ AUs the extents size = 64*1AUs

Figure 7 shows the Oracle ASM file extent relationship with allocation units. The first eight extents (0 to 7) are distributed on four Oracle ASM disks and are equal to the AU size. After the first 20000 extent sets, the extent size becomes 8*AU for the next 20000 extent sets (20000 - 39999). This is shown as bold rectangles labeled with the extent set numbers 20000 to 20007, and so on. The next increment for an Oracle ASM extent is 64*AU.

Figure 7. HP Scalable Warehouse Solution for Oracle – DL980 recommended 8Gb FC HBA Slot Loading



The automatic allocation of extent sizes was found to be an issue when performing testing and can likely cause issues on other storage arrays as well when very large ASM diskgroups are created. Large ASM diskgroups are typical for many Business Intelligence and data warehouse implementations. One way of avoiding this "growing" extent problem is to first create the diskgroup and then apply the Oracle ASM underscore parameter *_extent_counts* to negate extent growth. This can only be done after the diskgroup is created, and also needs to be done before any data is placed in the ASM diskgroup, to avoid potential performance issues. It is an attribute of this specific diskgroup needed to keep a consistent extent size.

When creating a disk group, add the "set attribute" clause to force fixed size striping. Variable extents sizes can be disabled on an ASM diskgroup after creation by issuing the following Oracle command.

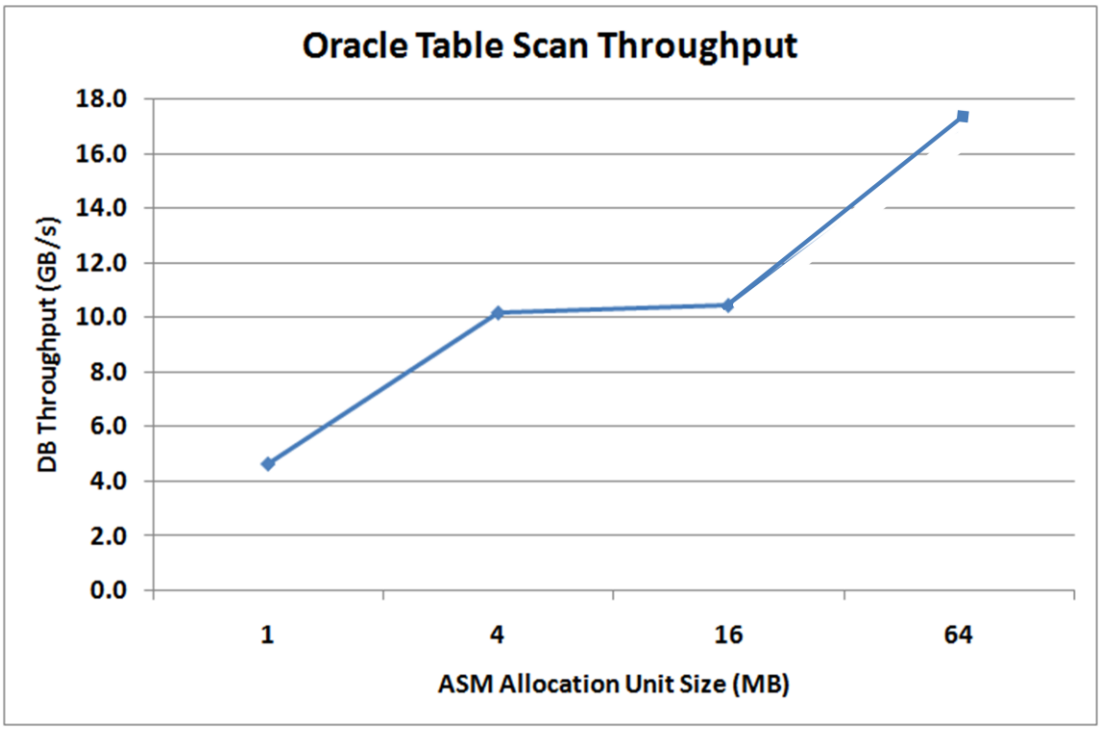
- `alter diskgroup <diskgroup name> set attribute '_extent_counts'='214748367 0 0';`

Next some testing was done to determine the optimum ASM AU size for data layout on the P2000 for BI/DW workloads. Tests were done using 1, 4, 16 and 64MB units of allocation as shown in figure 8. Clearly for our environment the uniform ASM AU size of 64MB achieved the best I/O results at nearly 18GB/sec for table scans. It was also found that using tablespace uniform extent size of 64MB and table uniform extent size of 64MB yielded the best results. However not nearly as dramatic a difference as it was with uniform ASM AU sizes, if the tablespace extent size was 16MB or greater (defined in increment factors of 2) it produced virtually identical results to 64MB

tablespace extent sizes. The table extent sizes didn't make nearly as much difference whether it was 1MB or larger, also using increments of power of 2 steps (1, 4, 16, or 64). The performance results were fairly similar.

Be aware that if using large table extents on tables with a large number of partitions the minimum table size can be very large, so it is best to use large extents for fact tables and very large dimension tables. For example if using 64MB extents for a table with 2500 range partitions would yield storage consumption at a minimum of 160GB (=2500*64MB) and if using composite partitions it could be drastically more consumption of storage space.

Figure 8. Results of Oracle table scan throughput using different ASM allocation unit sizes



Linux configuration details

When installing and configuring Red Hat Linux on the DL980 server please refer to the Red Hat Enterprise Linux 5 Installation Guide and the 5.7 Release Notes.

Enable NUMA mode

The 8-socket DL980 G7 with HP PREMA architecture performs very much like a traditional SMP system. The non-uniform memory access (NUMA) nature of the Xeon architecture is less pronounced than on many other enterprise platforms. Some minimal investigation of the Oracle database NUMA features was done as part of the work described in this white paper. We saw some improvement on small queries when enabling those features with "_enable_NUMA_support=TRUE". That setting did not have any discernable impact for larger, BI-type queries. Test your application workload to determine if NUMA will provide additional performance.

Use HUGE pages for SGA

The so-called "Huge page" feature of Linux is strongly recommended for all DL980 installations of Oracle. Please reference "HugePages on Oracle Linux 64-bit [ID 361468.1]" located on the Oracle support website – http://support.oracle.com.

Huge pages will save memory. For each user connecting to Oracle, the operating system creates a new copy of the SGA page table. With regular 4K pages, the OS page tables quickly grow bigger than the SGA itself. For example, an 800GB SGA with 1024 users would need 1.6TB of page tables. In practice the physical memory is limited, so the SGA would have to be scaled back to 266GB because the page tables would need 534 GB of memory.

By enabling huge pages it will eliminate this page table problem. The same example 800GB SGA with 1024 users now only requires 32GB for page tables. Even with thousands of users, HUGE pages keep the page tables manageable.

To make sure Oracle 11gR2 database can use Huge Pages in Red Hat Enterprise Linux 5, you also need to increase the ulimit parameter "memlock" for the oracle user in /etc/security/limits.conf unlimited.

To enable a 700GB SGA with HUGE pages, you will need to set the following parameters:

Table 10. Example of Linux and Oracle parameter settings for a 700GB Oracle SGA using HUGE pages

Init.ora

```
memory_target=0
sga_target = 0
sga_max_size= 740881858560
pga_aggregate_target = 200G
statistics_level = typical
use_large_pages = only
```

/etc/security/limits.conf

```
oracle soft memlock unlimited
oracle hard memlock unlimited
```

/etc/sysctl.conf

```
vm.nr_hugepages = 358500 #2M pages
kernel.shmmax = 751828992000 #bytes
kernel.shmall = 183552000 #4K pages
```

grep –i huge /proc/meminfo

```
HugePages_Total:   358500
HugePages_Free:    55508
HugePages_Rsvd:    50545
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

Setting up Device-Mapper Multipathing

Device-Mapper (DM) Multipath was used to enable the DL980 to route I/O over the multiple paths available to all the P2000 arrays. For P2000 arrays, HP is supporting DM Multipath which is bundled with the RHEL OS distribution or patch release. A path refers to the physical connection from an HBA port to the P2000 storage controller port. We used active/passive rather than load balancing or round robin multipathing options. Active/passive approach provided the best overall performance. When load balancing was tested the performance of the I/O throughput dropped significantly and we experienced higher I/O penalties. Please see Appendix 3 for an example of the multipath.conf file used in this test configuration.

With active/passive whenever an active path through which I/O happens fails, DM Multipath reroutes the I/O over other available paths. It provides transparent failover and failback of I/Os by rerouting I/Os automatically to this alternative path when a path failure is sensed, and routing them back when the path has been restored. For Linux on the DL980, when there are multiple paths to a storage controller, each path appears as a separate block device and hence results in multiple block devices for a single LUN. DM Multipath creates a new Multipath block device for those devices that have the same LUN WWN. Please refer to the Red Hat Enterprise Linux 5 DM Multipath configuration guide and the Native Linux Multipath Disk Arrays Device-Mapper for HP StorageWorks reference guide for more information on implementing multipathing (requires an HP Passport account).

Oracle Database configuration details

A standard installation of Oracle Database 11gR2 was performed on top of a standard installation of Red Hat Enterprise Linux 5.7. This white paper does not cover installation of the Oracle 11gR2 database. Anything specific from the engineering testing for this data warehouse reference architecture will be discussed.

For installing Oracle 11gR2 database please reference the following documents:

- Oracle Database Release Notes 11g Release 2 (11.2) for Linux
- Oracle Database Installation Guide 11g Release 2 (11.2) for Linux
- Oracle Database Quick Installation Guide 11g Release 2 (11.2) for Linux x86-64

Disable Oracle's Automatic Memory Management (AMM)

While Oracle's automatic memory management (AMM) promises to improve memory efficiency, AMM is incompatible with huge pages. The overhead of regular size page tables far exceeds any possible gains from AMM. Overall, AMM ends up wasting memory, and it should not be used for BI Data Warehousing workloads.

If you have configured your system for huge pages as described above, then AMM will already be disabled.

Oracle parallel query management in 11gR2

With 11gR2 Oracle introduced additional parallel query management features to improve BI/DW query performance. Parallel execution enables the application of multiple CPU and I/O resources to the execution of a single database operation. It reduces response time for data-intensive operations on large databases typically associated with a decision support system (DSS) and data warehouses. Parallel execution is designed to exploit large amounts of available hardware resources. Oracle parallel execution will benefit the DL980 and P2000 reference configuration for the following reasons:

- Large number of physical and logical cores
- Sufficient I/O bandwidth
- Sufficient memory to support additional memory-intensive processes, such as sorting, hashing, and I/O buffers

The first improved option is parallel query queuing. In past versions of Oracle database software a large query that needed to execute in parallel could run into a performance issue if all the parallel slaves on the system were already consumed by other queries. This large query would then either run serially as a single process (which can take a very long time to complete) or if the Oracle parameter *parallel_min_percent* was set the large query would almost always fail. For more detailed information on parallel queries and this issue please see Oracle white paper "Oracle Database Parallel Execution Fundamentals October 2010" and "Parallel Execution and Workload Management for an Operational Data Warehouse".

The second improved option is that Oracle can now use the in memory buffer cache for parallel execution, where in the past parallel slaves would use direct reads of data from disk. Traditional parallel processing by-passed the database buffer cache for most operations, reading data directly from disk (via direct path I/O) into the parallel query server's private working space (PGA). This meant that parallel processing rarely took advantage of the available memory other than for its private processing. This feature in Oracle 11gR2 is referred to as "In Memory Parallel Execution" where much larger systems like the DL980 with a large resource footprint or even a large DL980 cluster has a lot more data that can be cached for query use to improve performance. By having parallel query servers access objects via the database buffer cache they can scan data significantly faster than they can on disk.

To enable both parallel queuing and parallel slave caching the Oracle init.ora parameter *parallel_degree_policy* needs to be set to *auto*, by default this parameter is set to *manual* to disable both of these functions. To enable just the parallel queuing option keep *parallel_degree_policy* set to *manual*, but set *_parallel_statement_queuing* to *TRUE*. These values can be applied to an individual session level as well as globally for the entire environment. Also note that for small queries, parallel processing can be disabled by setting the *parallel_min_time_threshold* to a certain execution time in seconds. This is supposed to prevent small queries from consuming valuable parallel slave processes on the system and simply allows the small query to execute in serial, thus leaving the parallel processes available for larger queries. The Oracle cost based optimizer will predict the length of time in which the query will run and then make the decision on whether to run the particular query as serial or parallel. During our tests setting this time threshold parameter proved to be very unreliable and most small queries did execute in parallel rather than serial, even with changing the *parallel_min_time_threshold* settings. The only consistent and reliable way to execute these small queries was to either manually set the session where the query is to be executed to serial (parallel limit of 1) or to put a hint in the SQL statement { /*+PARALLEL(1) */}.

You can also set the number of parallel processes needed for large queries at the same time by providing hints in these SQL statements. Typical numbers of parallel slaves tested were 8, 16 or 32 per process for the larger complex queries. Next the parameter of *parallel_min_percent* = 25 was set, this means that at least 25% of the parallel slaves need to be present and available for the query to execute. Setting of this parameter in earlier database versions would have caused the query to fail if the resources were not available at the time the query was to be executed. However with Oracle 11gR2 parallel query execution the query will now wait in the queue until enough parallel slaves are available for it to execute. The query wait queue is a simple FIFO queue. Individual queries don't need to be tuned each time. Instead of the query failing or running in serial mode, Oracle will now put the query into a queue to wait for enough slaves before executing. The DOP management is running the same as before, but now you have the option of turning on queuing.

Note
If you are doing a proof of concept (POC) where typical testing is running single queries to see how they perform, it is recommended to set *parallel_degree_policy* to manual. Otherwise the performance of single queries may not be as good. The benefits of auto will be for multiple user concurrency type tests.

For the larger more complex queries we used the large 32k blocks for large fact tables (Oracle parameter db_32k_cache_size). This will allow fewer Oracle buffer blocks for the system to be managed when querying very large objects in the data warehouse. For smaller tables 8k block size was used.

Measured performance

During our performance testing, we used simulated sequential read workloads that are more "real world" than marketing specifications to gauge performance. These tests use workloads with varying sizes ranging from 100 to 500 concurrent queries, running simple, intermediate and advanced queries. From the results of these tests we determine both logical (how fast data can be read from buffer cache) and physical (how fast data can be read from physical disk) scan rates when turning on and off parallel caching. Table 11 below lists the observed scan rates for this configuration.

Table 11. Scan rate results for a customer realistic workload showing the results of using parallel caching

Test Scan Rate Type	Test Scan Rate	Comment
Test workload with parallel caching turned off	Scan time 8.19 sec. 17 GB/sec	Physical I/O
Test workload with parallel caching turned on	Scan time 3.93 sec. 35 GB/sec	Logical I/O
Average I/O	9.4 GB/sec	This is the average of the physical scan rates

Results of Parallel Query Execution testing

Single query testing – Parallel_cache tests

Tables scans were performed by calculating the sum of a single column in a large 1TB table with no predicate filtering, thus forcing the 11gR2 Oracle database to read the complete table every time (full table scan). Caching of any data in the db buffer blocks was turned off during this test. This test1 using 1TB table query scan resulted in no caching and the table was read from disk every time with the Oracle init.ora parameter *32k_cache_size* set to 550GB.

The OS utility vmstat shows read rate data of up to 17GB/s for complex Oracle query test1.

```
procs -----------memory----------          ---swap--   -----io---- --system-- -----cpu------
 r  b   swpd   free       buff   cache      si   so    bi     bo   in    cs   us sy id wa st
22 237 125248 201308192 4996324 72876672    0    0  17718496  292 92093 29623 14  3  7 77  0
24 236 125248 201150432 4996324 72876864    0    0  17197216   32 91338 28599 14  2  9 75  0
24 233 125248 201146736 4996324 72876864    0    0  16751136   68 90546 28281 14  2  6 78  0
34 226 125248 201145056 4996324 72876864    0    0  17154880   16 91658 27400 14  2  9 75  0
```

The same test1 query scan was executed again but this time turning caching on. The end results were very similar. Oracle 11gR2 determined that the table and data being read was much larger than the available cache and avoided trying to cache the table. There are parameters in Oracle 11gR2 to control Oracle's parallel query caching specific behavior.

Another test2 was done using a 138GB table. The first query scan produced around 17 GB/s throughput rate according to vmstat. This result was very similar to test1 but with a lot of I/O wait.

The scan time was 8.19 sec. or 17 GB/s. The same test2 query scan was executed again with parallel caching on. The results produced slightly lower I/O scan rates at 15GB/s according to vmstat. Also there was much higher user CPU consumption at 41% compared to 14% without parallel caching. The first test2 results had the processes just waiting in I/O wait state compared to much lower I/O waits for the second test2 test which was executed using the buffer cache.

The OS utility vmstat shows read rate data of up to 15GB/s for complex Oracle query test2.

```
procs -----------memory----------   ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff    cache   si   so    bi    bo   in   cs  us sy id wa st
39 222 125244 205236384 4996336 72880352    0    0 15248480     0 131463 44024 44  3  4  49  0
62 205 125244 205234144 4996336 72880352    0    0 14903900     8 128891 44026 41  3  5  51  0
```

Test2 scan time was 8.81 sec. or 15.6GB/s.

Test3 scan with caching on. The result was very low I/O (most data cached) less than 3GB/s. However the CPU usage was high at 99% for a short period of time.

```
procs -----------memory----------   ---swap-- -----io---- --system-- -----cpu------
r   b   swpd   free   buff    cache   si   so    bi    bo   in    cs  us sy id wa st
252  9 125244 205255616 4996336 72881184    0    0  2355766   44  87425  38368 42  1 57  0  0
249 11 125244 205240224 4996336 72881192    0    0  3023751    0  150553 46370 99  1  0  0  0
241  6 125244 205238688 4996336 72881192    0    0  1883235    8  146579 46369 99  1  0  0  0
```

Test3 scan time was 3.93 sec., half of test2 time.

A test4 query was run after reading a different large table and flushing the cache with different data. The new scan time was 9.24 sec. for the 138GB table. However in this case the parallel caching cannot be controlled, and often these large tables get flushed out, resulting in wasted buffer cycles. It is recommended to carefully test this feature for specific customer workloads to determine the actual benefit or penalty that can be incurred.

Multiple query testing – Parallel queuing tests with query concurrency

To test parallel queuing of a large user data (5TB) a large volume of 200 mixed concurrent queries (entry, intermediate, and advanced queries) were used. This test represents a more realistic picture of what can be expected in customer production environments.

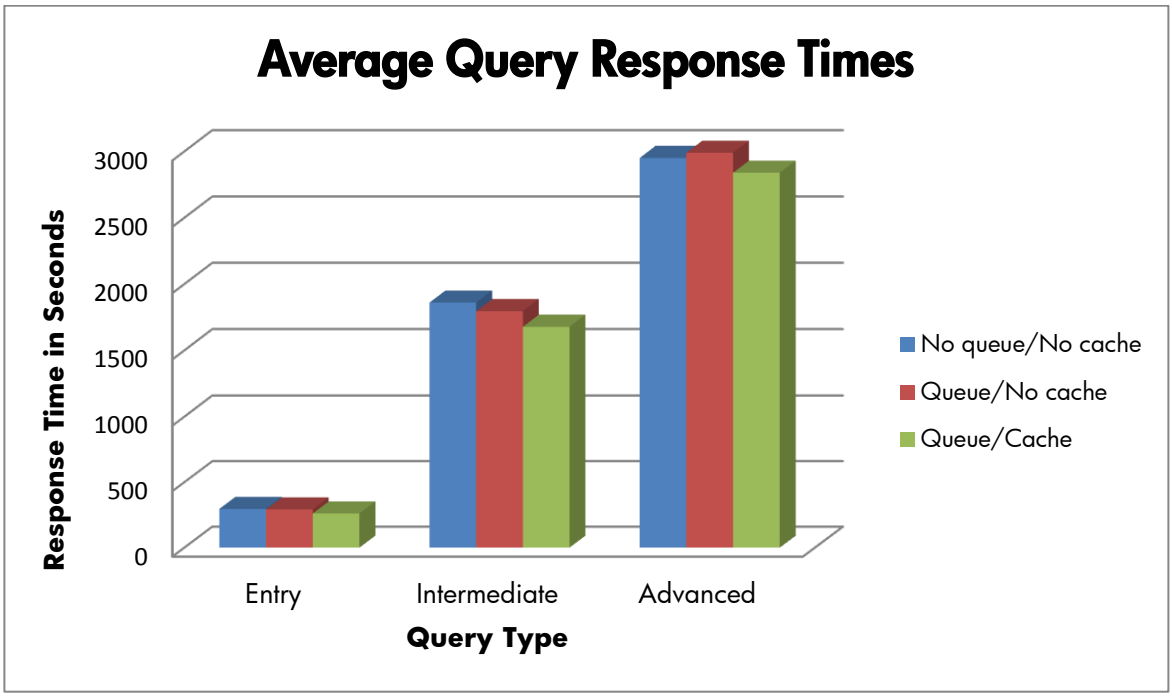
Entry level queries were run in serial by setting the degree of parallelism (DOP) to 1 at session level {alter session force parallel query parallel 1;}

The intermediate queries were set to use 16 DOP processes and queuing enabled by adding a hint parameter to the sql query code, { /*+PARALLEL(16) STMT_QUEUING */ }, also *parallel_min_percent* was set to 25% to force queuing if there are not enough parallel processes available.

The advanced queries were set to 32 DOP with the same hint parameter settings as intermediate queries ({ /*+PARALLEL(16) STMT_QUEUING */ }, and *parallel_min_percent* =25).

In the graph shown in figure 9 it can be seen that there is some additional benefit by enabling both queuing and parallel query caching of the queries when viewing the average query response times. Bigger benefits can be expected in more optimum conditions, where the database buffer cache size is much closer to the active data size and where there is lower concurrency. Most production workloads will only actively access a small portion of the total data in the database, and with increasing memory footprint of the systems it can lead to circumstances where a substantial amount of the active data can reside in the database buffer cache for BI type workloads.

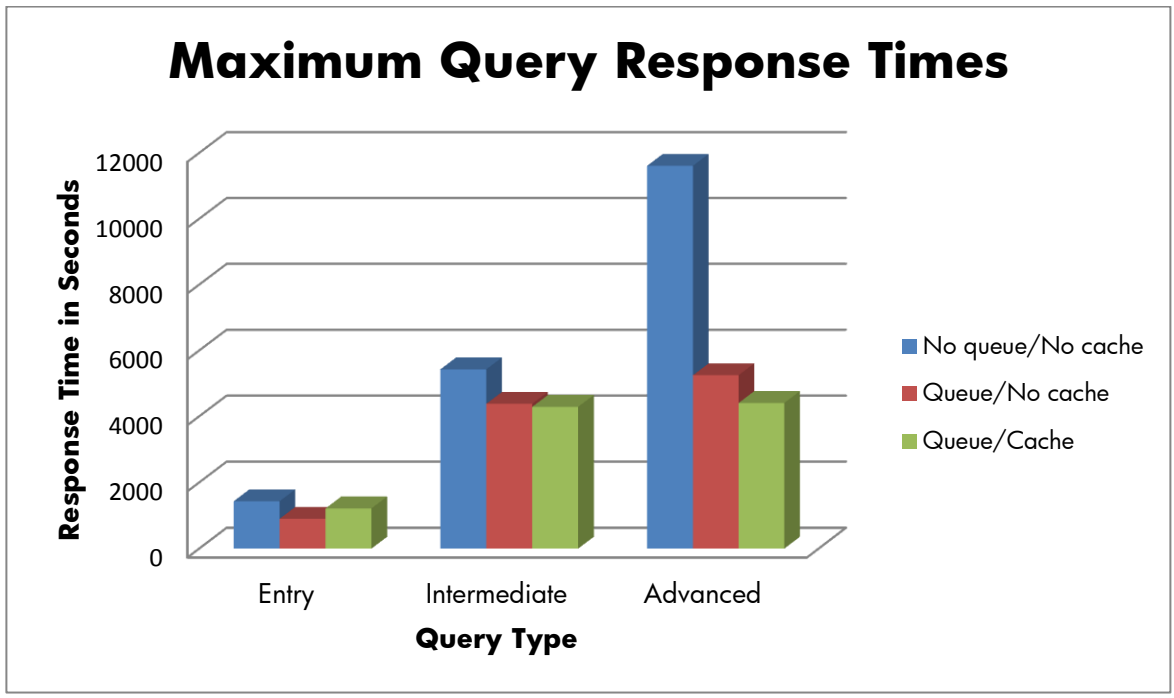
Figure 9. Average Query Response Time results of 200 mixed entry, intermediate and advanced queries by changing queuing and caching parameters



Entry level queries shows minimal improvement with both queuing and caching enabled.

However, note in figure 10 the difference it can make to the maximum response time a query can take to execute if queries are not queued. They end up executing in serial rather than parallel which is a single process that takes much longer time to complete larger complex queries. Alternatively large queries can be terminated if the minimum DOP requirement is not met.

Figure 10. Maximum Query Response Time results of 200 mixed entry, intermediate and advanced queries by changing queuing and caching parameters



If you review the Oracle Automatic Workload Repository (AWR) report when queries are being executed with the queuing option set, the following wait event will typically be observed in the Top 5 Timed Foreground Events. To see which of your queries are waiting in the queue the Oracle v$sql_monitor view can be queried for a STATUS='QUEUED'. You may notice a fairly long wait for direct path read. What it really means is that 8 queries were queued until enough parallel slaves were available to efficiently complete. The important takeaway is both that the AVERAGE response time goes down because the overall CPU consumption and I/O throughput goes UP.

Figure 11. Top 5 Timed Foreground Events you should observe when executing queries with queuing set

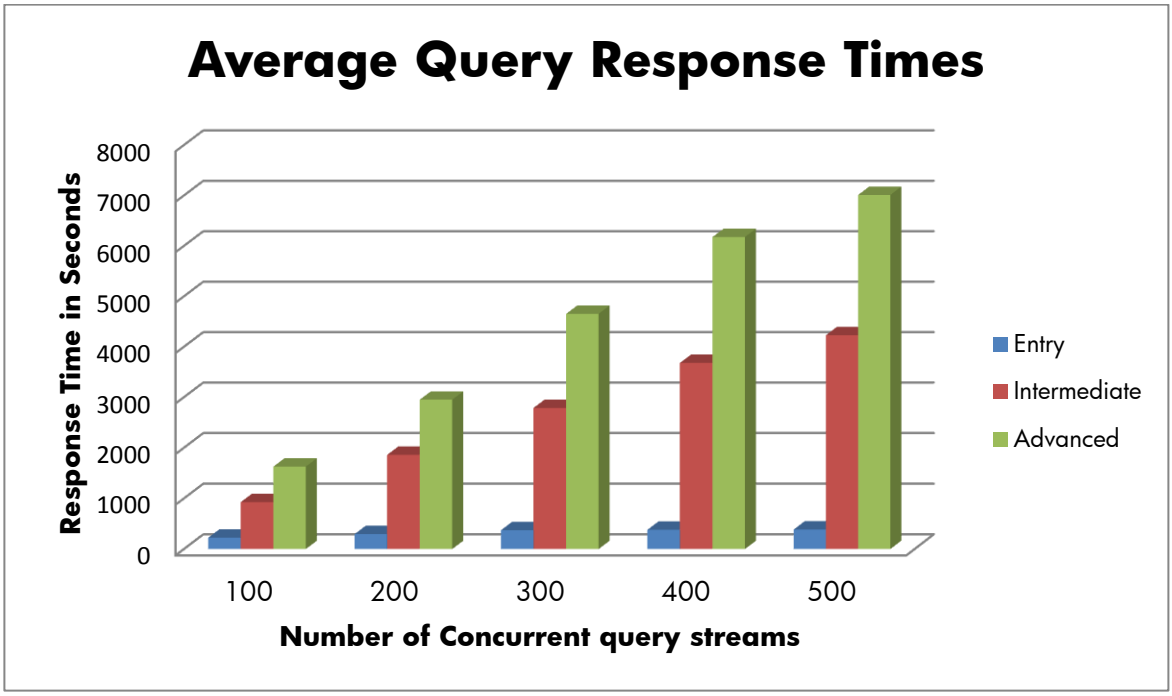
Event	Waits	Time(s)	Avg wait (ms)	% DB time	Wait Class
DB CPU		65,589		43.36	
resmgr:pq queued	8	31,366	3920695	20.74	Scheduler
direct path read	407,708	10,823	27	7.16	User I/O
read by other session	39,094	997	26	0.66	User I/O
db file scattered read	39,574	875	22	0.58	User I/O

For the last concurrent test the same 5TB of raw user data was used. No materialized views were used and this time all the data was actively queried. Tests were done in 100, 200, 300, 400 and 500 concurrent queries. The DL980/P2000 was able to consume most available resources when executing 100 concurrent queries all at the same time. This was due to the Oracle parallel management configuration set in the init.ora parameter file. You will notice as we increased the number of concurrent queries from 100 to 200, 300, 400 and 500 that the more complex intermediate and advanced query response times increased in a linear fashion (figure 12). Since most system resources were consumed with 100 concurrent queries the remaining queries were waiting in the parallel queue to be executed. You will also notice that the smaller queries executed in serial rather than parallel due to the *parallel_min_time_threshold* value being set to allow parallel execution to be disabled. The entry level queries had a less than linear increase in execution time as we scaled up concurrency due to the serial execution of these types of queries and limited dependency on parallel processes.

Keep in mind that these tests show the number of queries executing in parallel streams from query generators, which is far more taxing than an actual production environment where connected users would have some think time between issuing queries. The query generator will immediately issue a new query as soon as the previous query finishes, thus insuring a constant number of queries being issued to the system. In production environments connected users will not always have an active query working on the system as there is substantial think time between queries with analytic workloads.

The DL980/P2000 performed very well through all tests, but it is essential to size the memory correctly. Larger numbers of concurrent queries consume a lot more PGA memory that can force the system to begin to swap and eventually become unstable. This can be fixed by resizing the memory allocation for Oracle as shown in Appendix 2.

Figure 12. Average query response times for entry, intermediate and advanced queries as we increased the workload

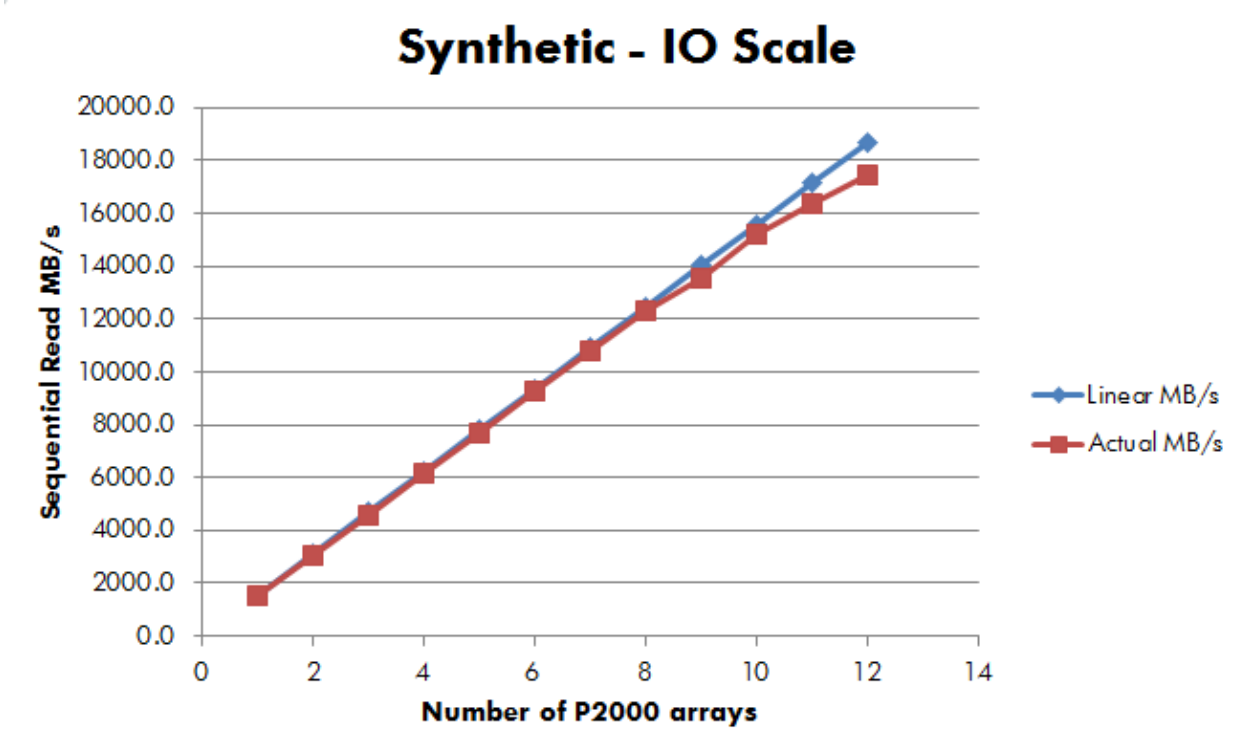


Synthetic I/O testing

To get a good indication about how well this particular reference configuration would scale for throughput, a set of low level I/O testing was conducted. Table 12 shows some low level test data that was collected when testing the I/O subsystem. An HP storage I/O tool for Linux was used to perform the synthetic I/O testing. Results were close to the theoretical 19GB/sec bandwidth of this configuration.

Table 12. Test results using HP storage I/O testing tool

P2000 Arrays	Linear MB/s	Actual MB/s
1	1559.2	1559.2
2	3118.3	3070.5
3	4677.5	4586.1
4	6236.6	6146.3
5	7795.8	7695.1
6	9355.0	9262.1
7	10914.1	10778.6
8	12473.3	12291.8
9	14032.5	13566.3
10	15591.6	15185.5
11	17150.8	16336.8
12	18709.9	17431.1



Best practices when deploying

Linux operating system

- The DL980 uses HP PREMA architecture which incorporates a new node controller design with Smart CPU caching and redundant system fabric. Combined with the Red Hat Enterprise Linux operating system, these features provide a solution that is fully capable of supporting the most demanding, data-intensive workloads, with the reliability, availability, and efficiency needed to run all of your business-critical applications with confidence. For more details on Linux and Oracle best practices see Best Practices When Deploying Linux on HP ProLiant DL980. This document also includes information about NUMA.
- Set HUGE pages for system global area (SGA) to 2MB. The main advantages of creating an SGA using huge pages has to do with increased performance by improving the translation lookaside buffer (TLB) hit ratio and reducing the memory footprint required for mapping the SGA.
- If you enable ccNUMA, create 5 extra huge pages per processor than the size needed for the Oracle instance. Otherwise Oracle DB instance may not start. 8*5= 40 extra huge pages.
- Device-Mapper Multipath – use failover and failback (active/passive) only and not load balancing options for the P2000 arrays.
- Verify CPU utilization and I/O interrupts are balanced across CPUs.
  - To verify interrupts are evenly distributed, examine the system statistics for interrupts:
    ```
    vi /proc/interrupts
    ```
  - To verify CPUs are evenly balanced:
    ```
    mpstat -A
    ```

DL980 server best practices

- Make sure BIOS settings are configured as listed in Appendix 1.
- Avoid the DL980's PCIe bus slot 1; this is a generation 1 narrow bandwidth slot, which performs much slower than all other I/O slots in the system
- Disable x2APIC Mode.
- Enable Hyper-Threading to maximize resource usage.
- For maximum performance install a large memory kernel, Linux 5.x supports up to 1TB but Linux 6.x will be able to support a much larger memory footprint. This can help minimize contention for the I/O subsystem.
- Spread the same size DIMM memory evenly across all memory cartridge sockets for maximum performance. For the 8-socket configuration install memory DIMMs across all memory sockets of the eight CPUs for optimum ccNUMA performance.
- Use only dual or quad rank DIMMs as they are faster than single rank DIMMs.
- Configure memory per DL980 best practices to minimize memory latencies as recommended in HP ProLiant DL980 G7 Server User Guide.
- Leave the I/O Q-depth at 32 per LUN with the QLogic HBAs. The Q-depth is set at the QLogic driver level with a Linux command.

Oracle Database best practices

- Enable NUMA mode by setting the Oracle database parameter:
  - `_enable_NUMA_support = true`
  - If NUMA is properly enabled, a message of the form *"NUMA system found and support enabled"* will be stored in the Oracle alert.log file.
- We recommend disabling Oracle Automatic Memory Management:
  - `SGA_TARGET = 0`
  - `MEMORY_TARGET = 0` (Oracle DB 11g)

  - `MEMORY_MAX_TARGET = 0` (Oracle DB 11g)

- Use the following guidelines as a starting point for sizing memory:
  - Size the Oracle System Global Area (SGA) at about 70% of total memory (about 600GB for database buffer cache with the remaining 100GB for the other SGA pools – see Oracle parameters in Appendix 2).
  - Size the Oracle Process Global Area (PGA) at about 20% of total memory.
  - That leaves about 10% for the Linux system management.

- A BI/DW type workload will do a lot more aggregating, sorting of data, joins, etc. than typically required for an OLTP environment so you will need a larger PGA
- Oracle 11g introduced a new, internal mutex mechanism. While the new mutex is more efficient than previous locks, it is essentially a spinlock and can consume lots of CPU when things go bad. It is important to eliminate mutex contention whenever it shows up in the AWR report. For more detail see Latch, mutex and beyond.
- Multi-block read count and block size for reads of 16MB for the larger tables using 32KB blocks yielded the best results. Small tables' multi-block read count was 4MB using 8KB block size.
- For bulk loading of data into the Oracle 11gR2 database it is recommended to turn off redo logging for better performance. If logging is required make sure the redo logs are sized large enough so that log switches are greater than 3-5 minutes. There is a tradeoff between data load speeds and database logging.

Estimating number of parallel execution servers

The number of parallel execution servers associated with a single operation is known as the degree of parallelism (DOP). Obviously, you want to use a lot of resources to reduce the response times, but if too many operations take this approach, the system may soon be starved for resources, as you can't use more resources than you have available. Oracle database has built in limits to prevent overload. This is where turning on Hyper-Threading can help provide additional logical cores to assist in processing. An 80 core DL980 will have 160 logical cores. Oracle will show cpu_count parameter as the total logical cores on the server.

- PARALLEL_MAX_SERVERS – set to 4 * number of logical cores. (4*160 = 640)
- PARALLEL_MIN_SERVERS – set to 2 * number of logical cores. (2*160 = 320)
- PARALLEL_SERVERS_TARGET – set half way between max and min servers (480)

Location of Oracle 11gR2 binaries

The Oracle database software should be installed separately from the storage used for the database instance itself. The recommended configurations are designed with the intent that the Oracle binaries be placed on the same drives as the OS. There is no performance benefit to separating the OS and Oracle binaries on different drives, though some customers will chose to do so to simplify system backup and maintenance operations.

Oracle database file location

For the HP P2000 storage configuration it is recommended that the Oracle database components, such as data, indexes, undo, temp and redo, should be managed with Oracle Automatic Storage Management (ASM) to stripe across the storage arrays. ASM should be used in "external redundancy" mode, since the P2000 arrays themselves will provide the RAID level protection for the databases.

Memory allocation for OS

In an Oracle 11gR2 setup it is recommended to use any extra memory available on the system for the system global area (SGA). This can improve I/O performance. Leave 10% of the memory available for the operating system.

Storage best practices

- For backup/recovery areas just plug other servers/storage into the existing SAN fabric.
- When configuring the P2000 cache the 2 MB read ahead was best. Do NOT use "super sequential". The problem is that Oracle database issues multiple sequential streams for queries and the P2000 is looking for a single sequential stream when that parameter is turned on, so the environment did not perform as well if super sequential settings were used.
- Use the multipath.conf file to align the volume names on the storage array with their presentation on the DL980 (for example, DATA05_A2). This will allow ready identification of underperforming storage units or failed controllers.

- Present volumes to the DL980 by presenting one primary path and one backup path. Do NOT use multiplexing.
- Stripe across the arrays using Oracle ASM AU 64MB stripe size. Make Oracle tablespace the same as the ASM AU stripe size of 64MB. Oracle tables can be smaller or similar size to tablespace extents.

Bill of materials

Figure 13 shows the recommended configuration for the Scalable Warehouse Solution for Oracle Databases.

Figure 13. HP Scalable Warehouse Solution for Oracle

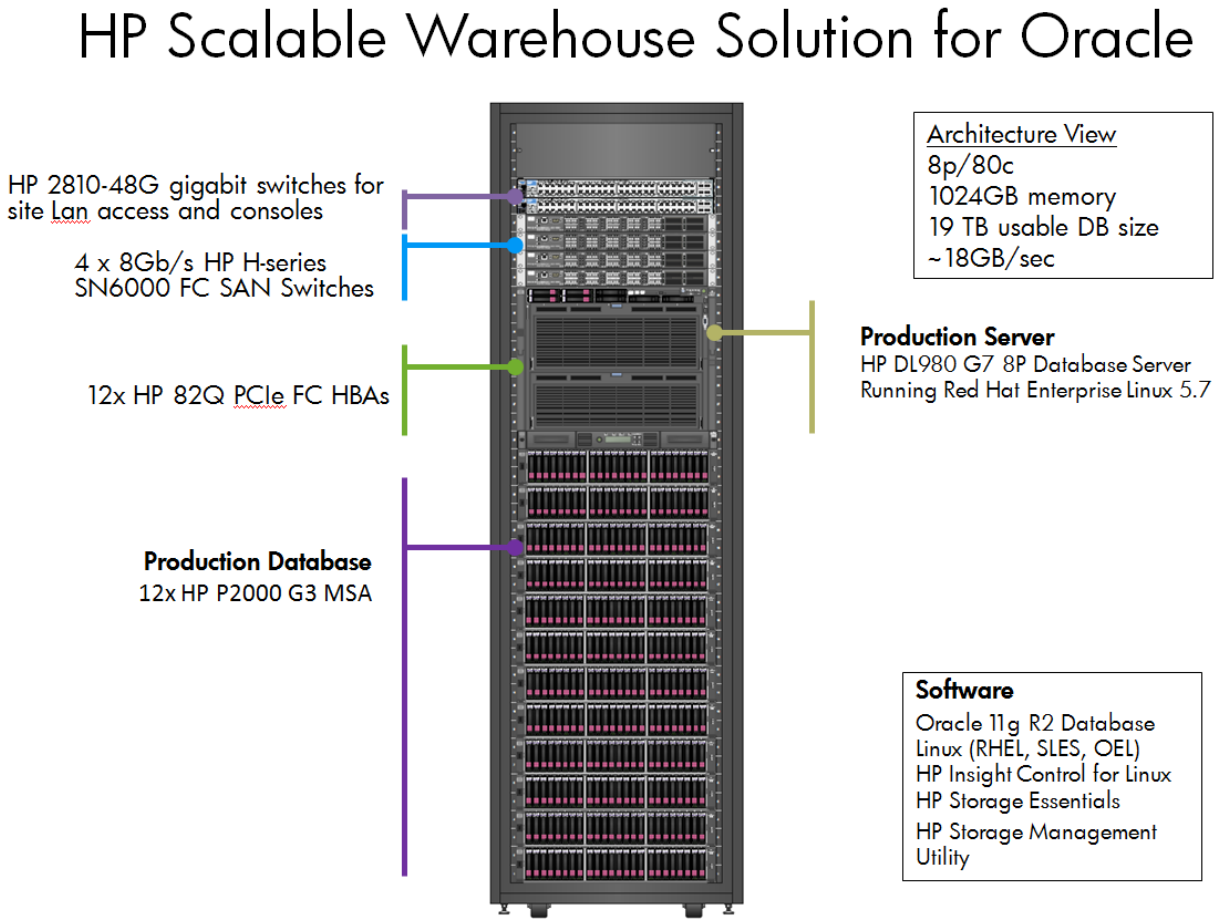

HP Scalable Warehouse Solution for Oracle
HP 2810-48G gigabit switches for site Lan access and consoles
4 x 8Gb/s HP H-series SN6000 FC SAN Switches
12x HP 82Q PCIe FC HBAs
Production Database
12x HP P2000 G3 MSA
Architecture View
8p/80c
1024GB memory
19 TB usable DB size
~18GB/sec
Production Server
HP DL980 G7 8P Database Server
Running Red Hat Enterprise Linux 5.7
Software
Oracle 11g R2 Database
Linux (RHEL, SLES, OEL)
HP Insight Control for Linux
HP Storage Essentials
HP Storage Management Utility

Table 13. HP Scalable Warehouse Solution for Oracle bill of materials

Qty	Description
	Production Database Server Configuration
1	HP ProLiant DL980 G7 CTO system
2	HP DL980 G7 E7-4870 FIO 4-processor Kit
1	HP DL980 CPU Installation Assembly
4	HP 1200W CS Platinum Power Supply kit
128	HP 8GB 2Rx4 PC3-10600R-9 kit
8	HP DL980 G7 Memory Board
1	HP Slim 12.7mm SATA DVD Optical kit
1	512MB Flash Backed Write Cache (FBWC)
1	PCI Express I/O Exp. Kit
1	Low Profile PCI Express I/O Expansion Kit
1	Dual Port 10GbE Server Adapter
4	HP 146GB 6G SAS 15K rpm SFF (2.5-inch) Dual Port Hard Drives
12	HP 82Q PCI-e 8Gb/s FC Dual Port HBA
	Storage Configuration
12	HP P2000 G3 Modular Smart Array Systems - 2.5-inch Drive Bay Chassis
24	HP P2000 G3 MSA Fibre Channel Dual Controller SFF
288	HP 146GB 6G SAS 15K rpm SFF Hard Drive
4	HP SN6000 Stackable 8 Gb 24-port Dual Power Fibre Channel Switch
24	Power Cord, 220/240 VAC
72	2m Multi-Mode Fibre Channel Cable
1	HP Storage Mirroring Linux Media Kit
1	HP Storage Mirroring Recover
1	42U rack assembly & relevant PDUs

Implementing a proof-of-concept

As a matter of best practice for all deployments, HP recommends implementing a proof-of-concept using a test environment that matches as closely as possible the planned production environment. In this way, appropriate performance and scalability characterizations can be obtained. For help with a proof-of-concept, contact an HP Services representative (http://www.hp.com/large/contact/enterprise/index.html) or your HP partner.

Appendix 1: BIOS and OS settings

The following were the settings of the indicated **BIOS parameters** used during these tests.

The BIOS version was P66 07/27/2010.

- System Options -> Processor Options -> Intel Hyperthreading Options -> Enabled
- System Options -> Processor Options -> Processor Core Disable -> All Cores Enabled
- System Options -> Processor Options -> Intel Turbo Boost Technology -> Enabled
- System Options -> Processor Options -> Intel VT-d -> Enabled
- Power Management Options -> HP Power Profile -> Maximum Performance
- Power Management Options -> HP Power Regulator -> HP Static High Performance Mode
- Power Management Options -> Redundant Power Supply Mode -> High Efficiency Mode (Auto)
- Power Management Options -> Advanced Power Management Options -> Minimum Processor Idle power Core State -> No C-states
- Power Management Options -> Advanced Power Management Options -> Minimum Processor Idle Power Package State -> No Package State
- Power Management Options -> Advanced Power Management Options -> Dynamic Power Savings Mode Response -> Fast
- Advanced Options -> Advanced System ROM Options -> Address Mode 44-bit -> Disabled (if system RAM is 1TB or less)
- Advanced Options -> Advanced System ROM Options -> x2APIC Options -> DISABLED (with RHEL5.6)
- Advanced Options -> Advanced Performance Tuning Options -> HW Prefetcher, Adjacent Sector Prefetcher and DCU Prefetcher -> all 3 options are ENABLED

The following kernel parameters were set in **sysctl.conf**

- `kernel.shmall = 183552000 # default is 2097152`
- `kernel.shmmax = 751828992000 # default = 4294967295`
- `kernel.panic_on_io_nmi = 1`
- `kernel.sem = 250 32000 100 128`
- `kernel.shmmni = 4096`
- `net.core.rmem_default = 262144`
- `net.core.wmem_default = 262144`
- `net.core.rmem_max = 4194304`
- `net.core.wmem_max = 1048576`
- `net.ipv4.ip_local_port_range = 9000 65500`
- `fs.aio-nr = 2000000`
- `fs.file-max = 6815744`
- `fs.aio-max-nr = 1048576`
- `vm.nr_hugepages = 358500`
- `vm.hugetlb_shm_group = 777`

The following limits were established for the *oracle* user in the **limits.conf**

- `oracle soft memlock unlimited`
- `oracle hard memlock unlimited`
- `oracle soft nproc 2047`
- `oracle hard nproc 16384`
- `oracle soft nofile 1024`
- `oracle hard nofile 65536`
- `oracle soft stack 10240`
- `oracle hard stack 256652256`

Appendix 2: Oracle parameter settings

The following Oracle parameters were used for these tests:

- compatible = 11.2.0.0.1
- _enable_NUMA_support = TRUE
- _parallel_statement_queuing = TRUE
- _pga_max_size = 4294967296
- control_files = (+DATA//control_001, +DATA//control_002)
- db_32k_cache_size = 590558003200
- db_block_size = 8192
- db_cache_size = 64424509440
- db_file_multiblock_read_count = 512
- db_keep_cache_size = 32212254720
- db_name = BI
- dispatchers = (PROTOCOL=TCP) (SERVICE=BIXDB)
- dml_locks = 13516
- java_pool_size = 3758096384
- large_pool_size = 3758096384
- lock_sga = TRUE
- open_cursors = 300
- log_buffer = 10048576
- parallel_adaptive_multi_user = FALSE
- parallel_automatic_tuning = FALSE
- parallel_degree_policy = auto
- parallel_execution_message_size = 32768
- parallel_force_local = FALSE
- parallel_io_cap_enabled = FALSE
- parallel_max_servers = 640
- parallel_min_percent = 0
- parallel_min_servers = 320
- parallel_servers_target = 480
- parallel_threads_per_cpu = 2
- pga_aggregate_target = 214748364800
- plsql_optimize_level=2
- processes = 2000
- recovery_parallelism = 0
- result_cache_max_result = 2
- result_cache_max_size = 5368709120
- result_cache_mode = manual
- sga_max_size = 740881858560
- shared_pool_size = 21474836480
- sessions = 3072
- statistics_level = typical
- timed_statistics = true

- `transactions = 3379`
- `undo_management = auto`
- `undo_retention = 900`
- `use_large_pages = only`

NOTE: Oracle parameters whose names start with an underscore character (e.g., "_in_memory_undo") are unsupported by Oracle. Extra care should be taken when changing the default setting of these so-called "underscore" parameters, especially in production environments. (The lone exception to this rule is the _enable_NUMA_support parameter, which is fully supported by Oracle.) When in doubt, please consult Oracle support.

Appendix 3: Example multipath.conf for P2000 array

```
### Do not edit the first two lines of this file or remove this file
### HP Device Mapper Multipath Enablement Kit v4.4.1
### The Device Mapper Multipath Template configuration file for RHEL5U4
### or later releases to be used with HP StorageWorks Arrays.
### Use this configuration file as your /etc/multipath.conf file.
### If you already have a valid working configuration file, refer here
### for the recommended configuration for HP arrays.
### For a list of configuration options with descriptions, please refer
### to /usr/share/doc/<multipath-tools-version>/multipath.conf.annotated

# The defaults section
defaults {
  udev_dir           /dev
  polling_interval   10
  selector           "round-robin 0"
  path_grouping_policy     failover
  getuid_callout           "/sbin/scsi_id -g -u -s /block/%n"
  #prio_callout            "/bin/true"
  prio                    "alua"
  path_checker             tur
  rr_min_io          100
  rr_weight          uniform
  failback           immediate
  no_path_retry            12
  user_friendly_names      yes
        uid                500
  gid                500
}

# The blacklist section - use this to blacklist a multipath device based on
# its wwid ( using wwid ) or device names ( using devnode ) or
# vendor and product id ( using device block).
blacklist {
# wwid        26353900f02796769
  devnode            "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
  devnode     "^hd[a-z][[0-9]*]"
# devnode            "^cciss!c[0-9]d[0-9]*"
# device {
#      vendor "HP"
#      product       "OPEN-*"
# }
}

# The blacklist_exceptions section - to list device names to be treated
# as multipath candidates even if they are on the blacklist.
# Note: blacklist exceptions are only valid in the same class.
#       It is not possible to blacklist devices using the devnode keyword
#       and to exclude some devices of them using the wwid keyword.

#blacklist_exceptions {
```

```
# wwid        2345234245647657
# devnode           "sda"
# device {
#      vendor "HP"
#      product      "OPEN-*"
# }
#}
# The multipaths section - uncomment this section to define a per multipath
# device settings.

#multipaths {
# uncomment the multipath block to modify the settings of a multipath device
# based on its wwid
#multipath {
# wwid              2039485769900000000
# alias                     red
# path_grouping_policy     group_by_prio
# path_selector             "round-robin 0"
# failback          immediate
# rr_weight         uniform
# no_path_retry             10
# rr_min_io         100
#}
#}
# The devices section - used to define per storage array model settings
devices {

# If you are already using a valid configuration file and do not have a
# device subsection for HP arrays, please add the appropriate device subsection
# for the respective arrays from the entries below.
# If you already have a device subsection for HP arrays which has different
# parameters from the entries below, modify it appropriately.

# For HP P2000 family
device {
        vendor                  "HP"
        product                 "P2000 G3 FC|P2000G3 FC/iSCSI"
        path_grouping_policy    group_by_prio
        getuid_callout          "/sbin/scsi_id -g -u -s /block/%n howdy"
        #getuid_callout          "/sbin/scsi_id --whitelisted --device=/dev/%n"
        path_checker            tur
        path_selector           "round-robin 0"
        #prio_callout            "/sbin/mpath_prio_alua /dev/%n"
        prio                    "alua"
        rr_weight               uniform
        failback                immediate
        hardware_handler        "0"
        no_path_retry           18
        rr_min_io               100
}

# To blacklist a device by vendor and product say, to blacklist a XP device
# uncomment the below block
```

```
#device {
# vendor                   "HP"
# product_blacklist "OPEN-.*"
#}


multipaths {
  multipath {
    wwid 3600c0ff000dabde80ef9f84c01000000
    alias  DATA_01_A1
  }
  multipath {
    wwid 3600c0ff000dabde816f9f84c01000000
    alias  DATA_01_A2
  }
  multipath {
    wwid 3600c0ff000daa9731ef9f84c01000000
    alias  DATA_01_B1
  }
  multipath {
    wwid 3600c0ff000daa97326f9f84c01000000
    alias  DATA_01_B2
  }
  multipath {
    wwid 3600c0ff000dabde858f9f84c01000000
    alias  FILES_01_A1
  }
  multipath {
    wwid 3600c0ff000dabde861f9f84c01000000
    alias  FILES_01_A2
  }
  multipath {
    wwid 3600c0ff000daa97369f9f84c01000000
    alias  FILES_01_B1
  }
  multipath {
    wwid 3600c0ff000daa97371f9f84c01000000
    alias  FILES_01_B2
  }
```

(Note: Continue this same configuration for user DATA and staging FILES on P2000 arrays 2-12)

```
  multipath {
    wwid 3600c0ff000dadaf7c404f94c01000000
    alias  DATA_12_A1
  }
  multipath {
    wwid 3600c0ff000dadaf7d104f94c01000000
    alias  DATA_12_A2
  }
  multipath {
    wwid 3600c0ff000dadc221905f94c01000000
```

```
    alias  DATA_12_B1
  }
  multipath {
    wwid 3600c0ff000dadc222605f94c01000000
    alias  DATA_12_B2
  }
  multipath {
    wwid 3600c0ff000dadaf7e805f94c01000000
    alias  FILES_12_A1
  }
  multipath {
    wwid 3600c0ff000dadaf7f505f94c01000000
    alias  FILES_12_A2
  }
  multipath {
    wwid 3600c0ff000dadc220506f94c01000000
    alias  FILES_12_B1
  }
  multipath {
    wwid 3600c0ff000dadc221706f94c01000000
    alias  FILES_12_B2
  }

}
```

Appendix 4: Oracle Advanced Compression

Many customers are looking for solutions that provide a means for reducing the size of their rapidly growing databases without negatively affecting their end user performance. Oracle 11gR2 offers integrated database compression to address this requirement.

We often think of compression as being a trade-off between performance and storage: compression reduces the amount of storage required, but the overhead of compressing and decompressing makes things slower. However, while there is always some CPU overhead involved in compression the effect on table scan I/O can be favorable, since if a table is reduced in size it will require fewer I/O operations to read it.

Prior to 11g, table compression could only be achieved when the table was created, rebuilt or when using direct load operations. However, in 11gR2, the Advanced Compression option allows data to be compressed when manipulated by standard DML (Data Manipulation Language). The data compression feature in Oracle 11gR2 Enterprise Edition reduces the size of tables and indexes while providing full row level locking for updates. There are two types of compression.

1. Row compression enables storing fixed-length data types in a variable-length storage format.
2. Page compression is a superset of row compression. It minimizes the storage of redundant data on the page by storing commonly-occurring byte patterns on the page once, and then referencing these values for respective columns.

Oracle's Advanced Compression offers three distinct levels: low, medium, and high. HP and Oracle recommend using the "low" method for best overall OLTP workload performance when data compression is desired. Oracle has provided a compression algorithm specifically designed to work with OLTP type workloads. This recommendation is based upon tests performed by HP and Oracle on industrial-standard x86 hardware (see the reference at the end of this document). Users may wish to evaluate other compression options to determine if the "medium" or "high" setting offers superior performance for their specific workload.

As one would expect, Oracle Advanced Data Compression was very effective at reducing disk utilization of traditional storage arrays. The result was improved large data scans from storage into the database instance for processing and reduced I/O wait overhead. Using table compression can reduce disk and memory usage, often resulting in better scale-up performance for read-only operations. Table compression can also speed up query execution by minimizing the number of round trips required to retrieve data from the disks. Compressing data however imposes a performance penalty on the load speed of the data. The overall performance gain typically outweighs the cost of compression. If you decide to use compression, consider sorting your data before loading it to achieve the best possible compression rate. Testing conducted by HP's Oracle Alliances team showed that Advanced Data Compression scaled linearly across the full range of CPU cores on HP 8-socket servers. The increases in CPU requirements from using Advanced Compression were around 5-10% for most tests.

For more information

For additional Oracle solutions from HP, please visit http://www.hp.com/go/oracle

For an overview of the HP ProLiant DL980 G7 server, http://www.hp.com/servers/dl980

HP ProLiant DL980 G7 server with HP PREMA Architecture, http://h20195.www2.hp.com/V2/GetDocument.aspx?docname=4AA3-0643ENW

ProLiant DL980 G7 server QuickSpecs, http://h18000.www1.hp.com/products/quickspecs/13708_div/13708_div.pdf

To download the recommended components described in this document, along with other drivers and software, visit the HP DL980 G7 support web page, http://www.hp.com/support/dl980g7

For a complete library of all documentation supporting the DL980 G7 server, http://www.hp.com/go/proliant_servers-docs (click on the link, "HP ProLiant DL980 G7 Server series")

HP P2000 G3 MSA Array Systems, http://www.hp.com/go/p2000

HP Single Point of Connectivity Knowledge (SPOCK) website, http://h20272.www2.hp.com

Open Source and Linux from HP, www.hp.com/go/linux

Learn more: www.hp.com/services/servers and www.hp.com/services/storage

Oracle Database Compression with HP DL785 and EVA: a scalability study, http://h20195.www2.hp.com/V2/GetDocument.aspx?docname=4AA1-0234ENW&cc=us&lc=en

If you do not know your HP sales or HP partner representative, please do not hesitate to send your response via email to one of the regional HP Oracle Solution Teams below.

Americas	ssa@hp.com
Europe / Middle East / Asia	oracle-sizing.emea@hp.com
Asia Pacific / Japan	oracle-sizing.apj@hp.com
Mexico	oracle-sizing.mexico@hp.com
Brasil	oracle-sizing.brasil@hp.com
Other Latin America / Caribbean	oracle-sizing.lac@hp.com

To help us improve our documents, please provide feedback at http://h71019.www7.hp.com/ActiveAnswers/us/en/solutions/technical_tools_feedback.html.

© Copyright 2011, 2012 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Intel and Xeon are trademarks of Intel Corporation in the U.S. and other countries. Microsoft is a U.S. registered trademark of Microsoft Corporation.

4AA3-8244ENW, Created November 2011; Updated February 2012, Rev. 1